Suite de commandes Git pour un flux de travail collaboratif sans écraser le travail de son prochain

Mouad – Maelainine – Wilfried

Octobre 2025

Résumé

Ce mémento fournit les commandes Git essentielles et pour notre projet. Il est conçu pour être simple et rapide à consulter.

1 Les Bases du Travail Quotidien

1.1 Se synchroniser : Tirer les dernières mises à jour

Le flux de travail local (Dossier sur son PC → Staging Area → Local Repository (dépôt local sur son PC). Avant de commencer tout travail, assurez-vous que **votre branche locale est mise à jour !**

1. `git status` : voir les différences avec la branche principale.
2. `git pull origin main` : Télécharger et fusionner toutes les modifications faites par les autres sur la branche principale. **C'est crucial**.

1.2 Créer une Branche de Fonctionnalité : Feature Branch

Pour travailler en parallèle, chacun doit isoler son travail de la branche principale stable. C'est l'étape la plus importante.

1. `git checkout -b nom-de-ma-branche` : Crée une nouvelle branche et y bascule immédiatement (-b pour branch).

Conseil : Nommez vos branches de manière descriptive (feature/connexion-utilisateur par exemple.)

1.3 Travailler et enrégistrer localement

Vous êtes maintenant sur votre branche isolée. Vous modifiez, ajoutez, supprimez des fichiers selon votre guise.

1. `git status` : Vérifier quels fichiers sont modifiés.
2. `git add .` : Préparer tous les fichiers modifiés pour le commit (les mettre dans la *Staging Area*).
3. `git commit -m "Ajout de la validation du formulaire de contact"` : Enregistrer les changements dans l'historique local de votre branche.

Note : Vous répéterez l'étape 3 autant de fois que nécessaire jusqu'à ce que la fonctionnalité soit complète.

1.4 Partager le Travail (Pousser)

Une fois que votre travail est complet ou que vous souhaitez le sauvegarder sur le (serveur distant).

1. `git push origin nom-de-ma-branche :` Envoie votre nouvelle branche et tous ses commits vers le dépôt distant.

1.5 Préparer la Fusion (Switch et Pull)

Tout en **restant** dans son dossier git local, dans un **cmd**, avant de fusionner votre travail dans main, il est bon de s'assurer que vous êtes **à jour** avec le reste de l'équipe.

1. `git checkout main :` Basculer (switch) vers la branche principale.
2. `git pull origin main :` Mettre à jour la branche principale avec les derniers commits des collègues.
3. `git checkout nom-de-ma-feature` : Revenir à votre branche de travail.

1.6 Fusionner votre travail

Vous êtes **actuellement sur votre branche** prêt à intégrer votre fonctionnalité dans la branche principale.

1. `git merge main` : Fusionne les derniers commit de main dans votre branche de travail. Ceci intègre les changements de vos collègues avant l'intégration finale.
2. Résoudre les conflits si Git le demande. Normalement, si toutes ces étapes précédentes sont suivies, il ne devrait pas y en avoir.
3. `git checkout main` : Revenir sur la branche principale.
4. `git merge nom-de-ma-branche` : Fusionne votre fonctionnalité complète dans la branche principale.
5. `git push origin main` : Pousse la branche principale mise à jour vers le serveur distant.

1.7 Nettoyage (Supprimer la branche)

Une fois la fonctionnalité intégrée et poussée, la branche de travail n'est plus nécessaire.

1. `git branch -d nom-de-ma-branche :` Supprime la branche **localement** (-d pour **delete**).
2. `git push origin -delete nom-de-ma-branche :` Supprime la branche **sur le dépôt distant (ou GitHub si vous préférez)**.

Le cycle **Pull → Branch → Commit → Push → Merge** est la méthode standard pour travailler en parallèle sans écraser le travail des autres.

2 Historique et Annulation (facultatif)

2.1 Voir l'Historique

— **Afficher l'historique des commits :** `git log`
— **Afficher l'historique de manière compacte (souvent préféré) :** `git log -oneline -graph -decorate`

2.2 Annulation et Correction

— **Annuler les modifications non stagées dans le Working Directory :** `git checkout -nom_du_fichier`
— **Déstager un fichier (le ramener de Staging à Working Directory) :** `git reset HEAD nom_du_fichier`
— **Modifier le dernier commit (Ajouter des changements ou corriger le message) :** `git commit -amend`