Table des matières

[Si PROBLEME = NOM DE FICHIER TROP LONG (PUSH ou Clone repositori) 2](#__RefHeading___Toc240_2903896080)

[ACTIVER PULL AUTOMATIQUEMENT 4](#__RefHeading___Toc242_2903896080)

[Annuler les synchronisations en attente (ou les pushes) qui n'ont pas encore été envoyées vers le dépôt distant 4](#__RefHeading___Toc271_2903896080)

[Augmentation taille du Buffer 7](#__RefHeading___Toc244_2903896080)

# Si PROBLEME = NOM DE FICHIER TROP LONG (PUSH ou Clone repositori)

**1. Exécuter PowerShell en tant qu'administrateur**

1. Fermez la fenêtre PowerShell actuelle.
2. Cliquez sur le bouton **Démarrer** et tapez "PowerShell".
3. Faites un clic droit sur "Windows PowerShell" et sélectionnez **Exécuter en tant qu'administrateur**.
4. Une fenêtre PowerShell avec des droits d'administrateur s'ouvrira.

**2. Exécuter la commande**

Dans la fenêtre PowerShell avec les droits administrateur, entrez la commande suivante :

git config --system core.longpaths true

Cette fois, la commande devrait s'exécuter correctement sans l'erreur de permission.

Cela permettra à Git de supporter les chemins longs sous Windows.

OU

**Modifier la configuration Git**

Vous pouvez essayer de configurer Git pour permettre des chemins longs avec la commande suivante dans Git Bash ou dans votre terminal :

git config --global core.longpaths true

Cette commande permet à Git de gérer des chemins de fichiers longs dans tous vos projets.

# ACTIVER PULL AUTOMATIQUEMENT

1. **Configurer Visual Studio Code pour une meilleure détection :**
   * Visual Studio Code ne notifie pas automatiquement des changements sur le dépôt distant. Toutefois, on peut configurer Git pour vérifier les mises à jour plus fréquemment. Pour ce faire, il faut modifier le paramètre git.autofetch dans Visual Studio Code :
     + Ouvre les **Paramètres** dans Visual Studio Code (Ctrl + ,).
     + Recherche git.autofetch.
     + Active cette option pour que Visual Studio Code fasse automatiquement un git fetch en arrière-plan à intervalles réguliers.

Visual Studio Code peut aider en récupérant régulièrement les modifications avec git.autofetch, mais tu devras toujours tirer (pull) manuellement pour intégrer les changements.

# Annuler les synchronisations en attente (ou les pushes) qui n'ont pas encore été envoyées vers le dépôt distant

vous pouvez utiliser différentes approches en fonction de votre besoin. Voici les principales façons d'annuler les synchronisations en attente dans Git.

**1. Annuler un git push en cours**

Si vous avez lancé un git push et que vous voulez l'annuler avant qu'il ne se termine, vous pouvez interrompre le processus avec Ctrl + C. Cependant, cela ne supprime pas les commits en attente localement, cela interrompt simplement le push en cours.

**2. Annuler les Commits Locaux en Attente**

Si vous avez des commits locaux que vous ne souhaitez pas synchroniser avec le dépôt distant, vous pouvez revenir en arrière et les supprimer ou les remettre en modification (staging).

**a. Revenir à l'état avant le dernier commit**

Pour annuler le dernier commit (mais conserver les modifications dans votre répertoire de travail), utilisez la commande suivante :

bash

Copier le code

git reset --soft HEAD^

Cela annulera le dernier commit mais conservera vos fichiers dans l'état modifié (staged) afin que vous puissiez soit les re-committer, soit les modifier.

**b. Annuler complètement le dernier commit et les changements associés**

Si vous voulez annuler le dernier commit **et** supprimer également les changements des fichiers associés, vous pouvez utiliser :

bash

Copier le code

git reset --hard HEAD^

Cela annulera le dernier commit et effacera les changements associés. **Attention :** Cela supprimera vos modifications non sauvegardées.

**c. Annuler plusieurs commits en arrière**

Si vous avez plusieurs commits à annuler, vous pouvez spécifier combien de commits revenir en arrière. Par exemple, pour annuler les 3 derniers commits :

bash

Copier le code

git reset --soft HEAD~3

Ou avec --hard pour effacer complètement les modifications des 3 derniers commits :

bash

Copier le code

git reset --hard HEAD~3

**3. Revenir à l'état avant le push**

Si vous avez déjà poussé des commits et que vous souhaitez annuler le push, vous devrez forcer un "rollback" sur le dépôt distant. Attention : cela peut impacter les autres développeurs qui utilisent ce dépôt, donc à utiliser avec précaution.

**a. Forcer un reset sur le dépôt distant**

Utilisez cette commande pour forcer le dépôt distant à revenir à un commit spécifique (par exemple, HEAD~1 pour revenir avant le dernier commit) :

bash

Copier le code

git push --force origin HEAD~1:main

Cela va réécrire l'historique sur le dépôt distant et annuler le commit poussé.

**4. Vérifier les commits en attente**

Avant d'annuler des commits, vous pouvez vérifier quels commits sont en attente de synchronisation avec le dépôt distant :

bash

Copier le code

git log origin/main..HEAD

Cela vous montrera tous les commits qui existent sur votre branche locale mais qui ne sont pas encore présents sur le dépôt distant.

**5. Annuler des modifications dans le "staging"**

Si vous avez ajouté des fichiers au "staging" avec git add mais que vous voulez annuler cela avant de faire un commit, utilisez :

bash

Copier le code

git reset HEAD <fichier>

Pour tout annuler du staging :

bash

Copier le code

git reset

Cela permet de retirer les fichiers du "staging" tout en conservant les modifications dans votre répertoire de travail.

**Résumé des Commandes Principales**

* **Annuler un push en cours :** Ctrl + C
* **Annuler un commit local :** git reset --soft HEAD^
* **Annuler un commit local et les modifications :** git reset --hard HEAD^
* **Annuler des modifications en "staging" :** git reset HEAD <fichier>
* **Forcer un reset sur le dépôt distant :** git push --force origin HEAD~1:main

# Augmentation taille du Buffer

Taper ligne de commande suivante : git config --global http.postBuffer 524288000

*Abandonner* tes changements locaux