# Commandes GIT

GIT est aujourd’hui le système de gestion le plus performant et pertinent.   
Il a été développé par Linus Torvalds en 2005.

Aujourd’hui, des millions d’entreprises l’utilisent pour gérer efficacement leur code source et la version de celui-ci.

Git est un logiciel open-source qui peut être téléchargé pour les plates-formes Linux, Windows et Mac.

# Commandes GIT de base

* Git config
* Git init
* Git add
* Git clone
* Git commit
* Git status
* Git push
* Git checkout
* Git remote
* Branche git
* Git pull
* Git merge
* Git diff
* Git tag
* Git log
* Git reset
* Git rm
* Git stash
* Git show
* Git fetch
* Git ls-tree
* Git cat-file
* Git grep
* Gitk
* Git instaweb
* Git gc
* git archive
* Git prune
* Git fsck
* Git rebase

# Commandes de base

## git config

L’une des commandes git les plus utilisées est git config.   
On l’utilise afin de configurer les préférences de l’utilisateur …  
adresse de courriel, l’algorithme utilisé pour diff, le nom d’utilisateur, le format de fichier, …  
**>> git config --global user.name <Nom d’utilisateur>  
>> git config --global user.name tux**

**>> git config --global user.email <Adresse de courriel>  
>> git config --global user.email tux@linux.local**

## git init

Cette commande est utilisée pour créer un nouveau dépôt GIT …  
**>> git init**

## git add

La commande **git add** peut être utilisée pour ajouter des fichiers à l’index (*stage*) …  
**>> git add <Fichiers|Répertoires>  
>> git add tux.texte  
>> git add \*.texte  
>> git add \*  
>> git add .  
>> git add ./documents**

## git clone

La commande **git clone** est utilisée pour la vérification des référentiels (dépôts).   
Si le référentiel se trouve sur un serveur distant …  
**>> git clone <Référentiel>  
>> git clone https://github.com/tux/Linux-base.git  
>> cd Linux-base**

**>> git clone <Référentiel> <Répertoire>  
>> git clone https://github.com/tux/Linux-base.git /tmp**

## git commit

La commande **git commit** permet de valider les modifications apportées la branche HEAD.   
Tout commit ne se fera pas vers le référentiel distant.  
**>> git commit --message="<Description|Message>"  
>> git commit -m "<Message>"  
>> git commit --message="2023-05-22"  
>> git commit -m "2023-05-22"**

## git status

La commande **git status** affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés.  
**>> git status**

## git push

**git push** est une autre commande GIT de base.   
Un simple push envoie les modifications locales apportées à la branche principale associée …  
**>> git push origin <master|main>**

## git remote

La commande **git remote** permet à un utilisateur de se connecter à un référentiel distant.   
Pour répertorier les férérentiels distants actuellement configurés …  
**>> git remote --verbose  
>> git remote -v**

Pour se connecter le référentiel local à un serveur distant …  
**>> git remote add origin <Adresse IP>  
>> git remote add origin 192.168.1.100**

## git pull

Pour fusionner toutes les modifications présentes sur le référentiel distant dans le répertoire de travail local, la commande pull est utilisée …  
**>> git pull**

## git diff

La commande **git diff** permet de lister les conflits.   
Pour visualiser les conflits d’un fichier …  
**>> git diff --base <Nom du fichier>  
>> git diff --base tux.texte**

Il est possible d’afficher les conflits entre les branches à fusionner, avant de les fusionner …  
**>> git diff <Nom de branche SOURCE> <Nom de branche CIBLE>  
>> git diff main test**

Pour simplement énumérer tous les conflits actuels …  
**>> git diff**

## git tag

Une balise est un objet faisant référence à un commit spécifique dans l'historique du projet, similaire aux marqueurs de chapitre dans un livre.  
Les balises sont utilisées afin de marquer des commits spécifiques avec des points d’ancrage (*handles*) simples.   
Pour baliser le dernier commit dans votre référentiel.  
**>> git tag <Balise>   
>> git tag 1.1.0**

## git log

L’exécution de cette commande génère le journal d’événements d’une branche.   
**>> git log <Identifiant du commit>   
>> git log a11b095…**

## git reset

Pour réinitialiser l’index et le répertoire de travail à l’état du dernier commit …  
**>> git reset --hard HEAD**

## git rm

Git rm peut être utilisé pour supprimer des fichiers de l’index et du répertoire de travail.   
**>> git rm <Nom du fichier>   
>> git rm tux.texte**

# Gestion des branches

La commande **git branch** peut être utilisée pour répertorier, créer ou supprimer des branches.   
Pour répertorier toutes les branches présentes dans le référentiel …  
**>> git branch**

Pour supprimer une branche …  
**>> git branch --delete <Nom de la branche>  
>> git branch -d <Nom de la branche>  
>> git branch --delete "test"  
>> git branch -b "test"**

## git checkout

La commande **git checkout** peut être utilisée pour créer des branches ou pour basculer d’une branche à l’autre  
Pour créer une branche …  
**>> git checkout -b "<Nom de branche>"  
>> git checkout -b "test"**  
Pour basculer d’une branche à une autre …  
**>> git checkout "<Nom de branche>"  
>> git checkout "test"**

## git merge

La commande **git merge** est utilisée pour fusionner une branche dans la branche active …  
**>> git merge "<Nom de branche>"  
>> git merge "test"**

## git diff

La commande **git diff** permet de lister les conflits.   
Il est possible d’afficher les conflits entre les branches à fusionner, avant de les fusionner …  
**>> git diff <Nom de branche SOURCE> <Nom de branche CIBLE>  
>> git diff main test**

## git stash

La commande **git stash** aide à enregistrer les modification qui ne doivent pas être commit immédiatement.   
C’est une sorte de commit temporaire.   
**>> git stash**

## git show

Pour afficher des informations sur tout fichier git, on utilise la commande **git show**.   
**>> git show**