Git Cheat Sheet: Commandes Essentielles & Bonnes Pratiques

Par Arthur Cartel Foahom Gouabou cartelgouabou.github.io • LinkedIn

GIT – CHEAT SHEET : Commandes Essentielles & Bonnes Pratiques

CONFIGURATION INITIALE

```
git config --global user.name "Votre Nom"
git config --global user.email "votre.email@example.com"
```

Définit votre identité pour tous vos dépôts Git.

INITIALISER UN PROJET

```
git init
```

Initialise un dépôt Git dans le dossier courant.

AJOUTER & ENREGISTRER DES FICHIERS

```
git add nom_fichier
git add .                # Tout ajouter
git commit -m "Message clair"
```

`add` prépare les fichiers, `commit` les enregistre dans l'historique.

VÉRIFIER L'ÉTAT

```
git status
```

Affiche les fichiers modifiés, suivis, non suivis, etc.

HISTORIQUE DES COMMITS

```
git log
```

Liste les commits avec auteur, date et message.

TRAVAIL AVEC DÉPÔT DISTANT

```
git remote add origin git@github.com:user/mon_depot.git
```

Relie un dépôt local à GitHub via SSH.

```
git push -u origin master   # Premier push
git push                    # Ensuite
```

Envoie vos commits sur le dépôt distant.

```
git pull
```

Récupère les modifications du dépôt distant.

COMPARER LES MODIFICATIONS

```
git diff
```

Compare les modifications locales avec la dernière version commitée.

IGNORER DES FICHIERS

Fichier `.gitignore` :

```
venv/
__pycache__/
*.log
```

Empêche Git de suivre certains fichiers ou dossiers.

ANNULER DES MODIFICATIONS

```
git restore fichier.py              # Annule une modif non committée
git reset --soft HEAD~1             # Annule le dernier commit (garde les modifs)
git reset --hard HEAD~1             # Supprime tout (dangereux)
```

🔥 GÉRER UN HOTFIX

🎯 Use Case :

Un bug critique est détecté en production. On doit corriger ça immédiatement sans attendre la prochaine release.

```
git checkout main
git pull
git checkout -b hotfix/nom_du_bug

# Corriger le bug, puis :
git add .
git commit -m "fix: corrige le bug critique X"
git checkout main
git merge hotfix/nom_du_bug
git push origin main

# Supprimer la branche hotfix
git branch -d hotfix/nom_du_bug
git push origin --delete hotfix/nom_du_bug
```

💡 Les hotfix sont traités comme des patchs prioritaires en dehors du cycle normal de développement.

📆 ROUTINE QUOTIDIENNE DE TRAVAIL

```
# Tous les matins
git checkout main
git pull origin main
```

💡 Récupérer les dernières modifications pour être à jour.

```
# Le soir (ou en fin de session)
git add .
git commit -m "WIP: avancement de la journée"
git push
```

💡 Sauvegarder votre travail pour ne rien perdre, même si ce n’est pas encore terminé.

✨ DÉMARRER UNE NOUVELLE FONCTIONNALITÉ

```
# Se mettre à jour
git checkout main
git pull origin main

# Créer une branche dédiée
git checkout -b feat/nom_fonctionnalite
```

💡 Utilisez des noms clairs et explicites pour les branches.

```
# Après le développement :
git add .
git commit -m "feat: ajoute la fonctionnalité Y"
git checkout main
git merge feat/nom_fonctionnalite
git push

# Supprimer la branche locale
git branch -d feat/nom_fonctionnalite
git push origin --delete feat/nom_fonctionnalite
```

☑ CONSEIL : COMMITEZ SOUVENT

Plus vos commits sont fréquents et clairs, plus il est facile de collaborer et de revenir en arrière si besoin.

Document rédigé par Arthur Cartel Foahom Gouabou – Formateur & Data Scientist • https://cartelgouabou.github.io/ • LinkedIn