GitHub

# AIDE-MÉMOIRE GITHUB GIT

V1.1.1

Git est le sytème de gestion de version décentralisé open source qui facilite les activités GitHub sur votre ordinateur. Cet aide-mémoire permet un accès rapide aux instructions des commandes Git les plus utilisées.

## INSTALLER GIT

GitHub fournit des clients desktop qui incluent une interface graphique pour les manipulations les plus courantes et une "an automatically updating command line edition of Git" pour les scénari avancés.

**GitHub pour Windows**
https://windows.github.com

**GitHub pour Mac**
https://mac.github.com

Les distributions de Git pour Linux et les systèmes POSIX sont disponibles sur le site web officiel de Git SCM.

**Git pour toutes les plate-formes**
http://git-scm.com

## CONFIGURATION DES OUTILS

Configurer les informations de l'utilisateur pour tous les dépôts locaux

```
$ git config --global user.name "[nom]"
```

Définit le nom que vous voulez associer à toutes vos opérations de commit

```
$ git config --global user.email "[adresse email]"
```

Définit l'email que vous voulez associer à toutes vos opérations de commit

```
$ git config --global color.ui auto
```

Active la colorisation de la sortie en ligne de commande

## CRÉER DES DÉPÔTS

Démarrer un nouveau dépôt ou en obtenir un depuis une URL existante

```
$ git init [nom-du-projet]
```

Crée un dépôt local à partir du nom spécifié

```
$ git clone [url]
```

Télécharge un projet et tout son historique de versions

## EFFECTUER DES CHANGEMENTS

Consulter les modifications et effectuer une opération de commit

```
$ git status
```

Liste tous les nouveaux fichiers et les fichiers modifiés à commiter

```
$ git diff
```

Montre les modifications de fichier qui ne sont pas encore indexées

```
$ git add [fichier]
```

Ajoute un instantané du fichier, en préparation pour le suivi de version

```
$ git diff --staged
```

Montre les différences de fichier entre la version indexée et la dernière version

```
$ git reset [fichier]
```

Enleve le fichier de l'index, mais conserve son contenu

```
$ git commit -m "[message descriptif]"
```

Enregistre des instantanés de fichiers de façon permanente dans l'historique des versions

## GROUPER DES CHANGEMENTS

Nommer une série de commits et combiner les résultats de travaux terminés

```
$ git branch
```

Liste toutes les branches locales dans le dépôt courant

```
$ git branch [nom-de-branche]
```

Crée une nouvelle branche

```
$ git checkout [nom-de-branche]
```

Bascule sur la branche spécifiée et met à jour le répertoire de travail

```
$ git merge [nom-de-branche]
```

Combine dans la branche courante l'historique de la branche spécifiée

```
$ git branch -d [nom-de-branche]
```

Supprime la branche spécifiée

# AIDE-MÉMOIRE GITHUB GIT

## CHANGEMENTS AU NIVEAU DES NOMS DE FICHIERS

Déplacer et supprimer des fichiers sous suivi de version

```
$ git rm [fichier]
```

Supprime le fichier du répertoire de travail et met à jour l'index

```
$ git rm --cached [fichier]
```

Supprime le fichier du système de suivi de version mais le préserve localement

```
$ git mv [fichier-nom] [fichier-nouveau-nom]
```

Renomme le fichier et prépare le changement pour un commit

## EXCLURE DU SUIVI DE VERSION

Exclure des fichiers et chemins temporaires

```
*.log
build/
temp-*
```

Un fichier texte nommé `.gitignore` permet d'éviter le suivi de version accidentel pour les fichiers et chemins correspondant aux patterns spécifiés

```
$ git ls-files --other --ignored --exclude-standard
```

Liste tous les fichiers exclus du suivi de version dans ce projet

## ENREGISTRER DES FRAGMENTS

Mettre en suspens des modifications non finies pour y revenir plus tard

```
$ git stash
```

Enregistre de manière temporaire tous les fichiers sous suivi de version qui ont été modifiés ("remiser son travail")

```
$ git stash pop
```

Applique une remise et la supprime immédiatement

```
$ git stash list
```

Liste toutes les remises

```
$ git stash drop
```

Supprime la remise la plus récente

## VÉRIFIER L'HISTORIQUE DES VERSIONS

Suivre et inspecter l'évolution des fichiers du projet

```
$ git log
```

Montre l'historique des versions pour la branche courante

```
$ git log --follow [fichier]
```

Montre l'historique des versions, y compris les actions de renommage, pour le fichier spécifié

```
$ git diff [premiere-branche]...[deuxieme-branche]
```

Montre les différences de contenu entre deux branches

```
$ git show [commit]
```

Montre les modifications de métadonnées et de contenu inclues dans le commit spécifié

## REFAIRE DES COMMITS

Corriger des erreurs et gérer l'historique des corrections

```
$ git reset [commit]
```

Annule tous les commits après `[commit]`, en conservant les modifications localement

```
$ git reset --hard [commit]
```

Supprime tout l'historique et les modifications effectuées après le commit spécifié

## SYNCHRONISER LES CHANGEMENTS

Référencer un dépôt distant et synchroniser l'historique de versions

```
$ git fetch [nom-de-depot]
```

Récupère tout l'historique du dépôt nommé

```
$ git merge [nom-de-depot]/[branche]
```

Fusionne la branche du dépôt dans la branche locale courante

```
$ git push [alias] [branche]
```

Envoie tous les commits de la branche locale vers GitHub

```
$ git pull
```

Récupère tout l'historique du dépôt nommé et incorpore les modifications