git-cmds-base

Alice commence à travailler avec git

Paramétrage et initialisations

On peut commencer par configurer quelques réglages globaux comme son nom, son email, ou la branche par défaut (normalement main depuis 2020) :

```
alice> git config --global user.name "Alice Torvalds"
alice> git config --global user.email "alice@kernel.org"
alice> git config --global init.defaultBranch main
```

puis lister ses réglages globaux pour vérification :

```
alice> git config -l
```

Création d'un dossier local versionné

```
mkdir monprojet
cd monprojet
git init
```

ou directement :

```
git init monprojet
```

Ajout de contenu dans monprojet

```
echo "# Mon projet" > Readme.md
git add Readme.md
```

On commite

```
git commit -m "ajout Readme.md"
```

On vérifie l'historique

```
git log
```

ou

```
git log --oneline
```

Ajout d'un nouveau fichier par Alice

```
echo "# Installation" > install.md
```

qui crée un nouveau fichier install.md

puis git status, add et commit :

```
alice> git status
alice> git add install.md
alice> git commit -m "Ajout install.md"
```

Modification du fichier Readme.md

```
    echo "[Installation](install.md)" >> Readme.md
```

on peut faire à nouveau un add de Readme.md puis un commit ou plus simplement :

```
git commit -am "Modif Readme.md"
```

qui réalise l'ajout et le commit simultanément.

git status et git show

Si vous voulez savoir où vous en êtes, n'oubliez jamais la commande magique :

```
alice> git status
```

et pour avoir des détails sur la contribution d'un commit, faire un `git show` sur son id (sha1, prononcez chaone, les premiers chiffres suffisent), par exemple :

```
git show d2ee25
```

Alice crée un dépot (privé) ”monprojet” sur Gitlab

Dépôts Gitlab et alternatives

Les dépôts sur Gitlab peuvent être :

- privés

- publics
- internes à gitlab

On peut aussi en créer sur :

Bitbucket
ou
Github

Connexion entre le local et le gitlab

```
alice> git remote add origin https://gitlab.com/alice/monprojet.git
```

Alice pousse son master sur son remote

```
alice> git push -u origin main
```

puis ultérieurement :

```
alice> git push
```

```
alice> git push --follow-tags
```

pour envoyer aussi les tags concernés dans le push

Autres réglages git

Mémoriser les identifiants

Sous Linux :

```
alice> git config --global credential.helper 'cache --timeout=7200'
```

mémorise vos identifiants pendant 2h

Sous Windows, gitbash est configuré par défaut avec le manager :

```
alice> git config --global credential.helper manager
```

le manager gère dans ce cas les identifiants.

Configurer son éditeur favori

```
alice> git config --global core.editor emacs
```

ou

```
alice> git config --global core.editor "code --wait"
```

Configurer sa stratégie de pull

Vous pouvez voir ce message apparaitre lorsque vous faites un `git pull`{.bash} :

```
_
 git pull
warning: Tirer sans spécifier comment réconcilier les branches ↩
   divergentes
est découragé. Vous pouvez éliminer ce message en lançant une des
commandes suivantes avant votre prochain tirage :

  git config pull.rebase false  # fusion (stratégie par défaut)
  git config pull.rebase true   # rebasage
  git config pull.ff only       # avance rapide seulement

Vous pouvez remplacer "git config" par "git config --global" pour que
ce soit l\'option par défaut pour tous les dépôts. Vous pouvez aussi
passer --rebase, --no-rebase ou --ff-only sur la ligne de commande pour
remplacer à l\'invocation la valeur par défaut configurée.
```

Le premier choix sera souvent privilégié, le 2ème uniquement si vous voulez procéder par rebase, cf ci-dessous et le 3ème va notamment refuser de merger si le local et le distant on divergé, ce qui peut-être sécurisant pour un débutant Git, et éviter de faire un commit de merge, ce qui présentera toutefois une difficulté si vous souhaitez annuler le merge ultérieurement.

On pourra donc généralement choisir :

```
git config --global pull.rebase false
```

ou

```
git config --global pull.ff only
```

sauf si on sait ce qu'on fait, quitte à revenir au cas par cas dans certains projets sur ce réglage.

Fichier .gitconfig

Il se trouve généralement dans le HOME de l'utilisateur, à la racine, une configuration typique avec vscode :

```
[user]
    name = Linus Torvalds
    email = linus@git.edu
[diff]
    tool = default-difftool
[difftool "default-difftool"]
    cmd = code --wait --diff $LOCAL $REMOTE
[push]
    default = simple
   followTags = true
[core]
    editor = code --wait
[color]
    ui = auto
[credential]
    helper = cache --timeout=7200
[merge]
    tool = vscode
[mergetool "vscode"]
    cmd = code --wait $MERGED
```

Corriger des erreurs

Défaire le dernier commit

Plusieurs solutions possibles mais la plus simple est :

```
git reset HEAD^
```

seul le commit est enlevé, le code est toujours présent.

Supprimer une révision

```
git revert 32ee587
```

supprime la révision indiquée de l'historique tout en créant un nouveau commit inversant les changements effectués auparavant