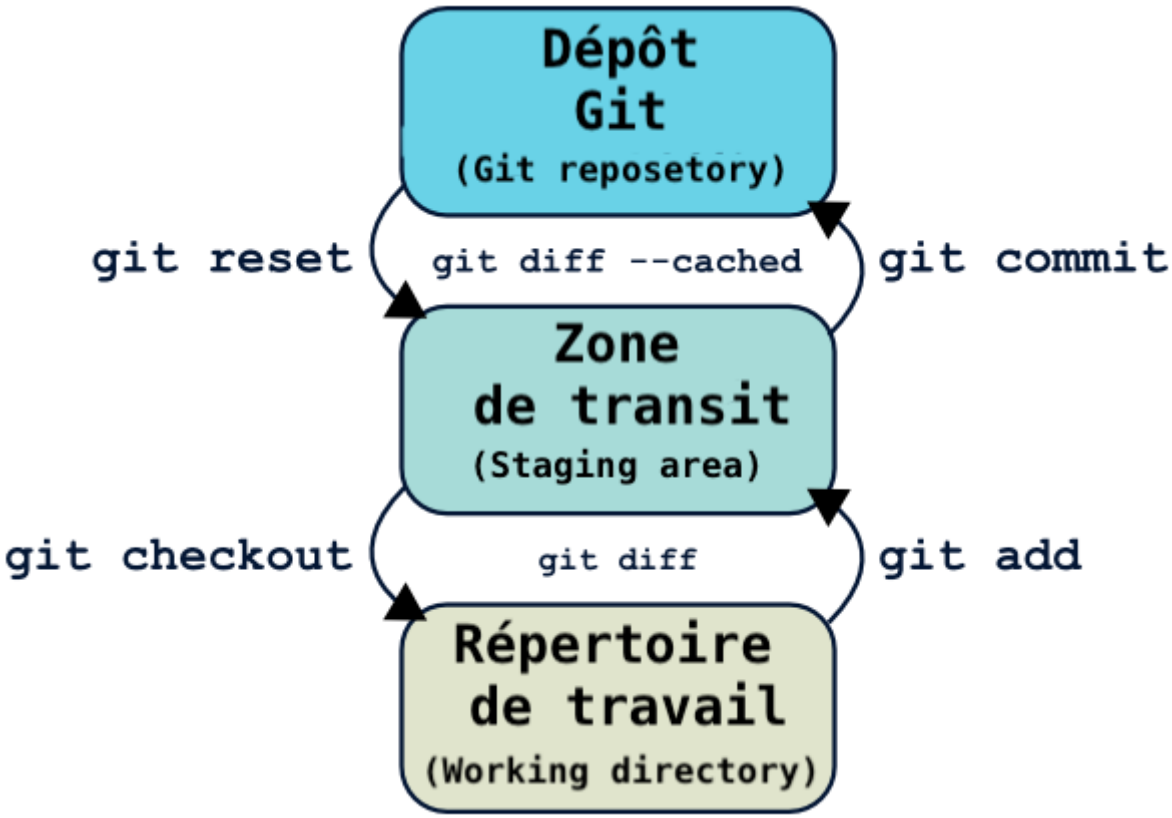
Commandes principales de Git

1. Introduction au processus de commit



Le processus de commit avec Git est le suivant :

- je développe en modifiant / déplaçant / supprimant des fichiers ;
- quand une série de modification est cohérente et digne d'être commitée, je la place dans la zone de transit (stagging) avec le commande git add ;
- je vérifie que l'état de ma zone de transit est satisfaisant avec git status ;
- je committe git commit ;

Pour copier un fichier du dépôt Git vers la zone de transit, on utilise git reset.

Pour copier un fichier de la zone de transit vers le repertoire de travail (donc supprimer les modifications en cours), on utilise git checkout.

Pour visualiser les modifications entre le répertoire de travail et la zone de transit, on utilise git diff.

Pour visualiser les modifications entre la zone de transit et le dernier commit, on utilise git diff –cached.

2. Personnaliser

Définir son identité :

```
git config --global user.email "email@example.com"
git config --global user.name "NOM Prénom"
```

Voir la configuration :

```
git config --list
```

3. Créer

Créer un nouveau dépôt local (dans le dossier courant) :

```
git init
```

Cloner un dépot existant :

```
git clone git://github.com/utilisateur/depot.git
```

4. Modifications locales

Afficher les fichiers modifiés dans votre répertoire de travail :

```
git status
```

Afficher les modifications sur les fichiers suivis :

```
git diff
```

Ajouter tous les changements actuels au prochain commit :

```
git add
```

Ajouter tous les changements de toute l'arborescence du projet :

```
git add --all
```

Commiter tous les changements locaux des fichiers suivis :

```
git commit -a
```

Commiter les modifications en attente :

```
git commit -m 'Votre message'
```

Il est possible de combiner les deux commandes :

```
git commit -am 'Votre message'
```

Modifier le commit précédent :

```
git commit --amend
```

5. Historique de Commit

Afficher tous les commits :

```
git log
```

Afficher tous les commits (uniquement l'identifiant et le texte) :

```
git log --oneline
```

Afficher l'historique d'un utilisateur uniquement :

```
git log --author="utilisateur"
```

Afficher l'historique des modifications pour un fichier uniquement :

```
git log -p <fichier>
```

Affiche les changements (en détails) dans le fichier :

```
git blame <file>
```

6. Branches & Tags

Lister toutes les branches :

```
git branch
```

Changer de branche :

```
git checkout <votre-branche>
```

Créer une nouvelle branche en se basant sur le HEAD :

```
git branch <votre-branche>
```

Créer une nouvelle branche de suivi, basée sur une branche distante :

```
git branch --track <nouvelle-branche> <branche-distante>
```

Supprimer une branche :

```
git branch -d <votre-branche>
```

Marquer le commit courant avec un tag :

```
git tag <non-du-tag>
```

7. Merge & Rebase

Fusionner la branche avec la master :

```
git checkout master
git merge <votre-branche>
```

⚠ Attention ⚠

Jouer avec l'historique est toujours dangereux surtout si vous travaillez à plusieurs !

Mettre à jour votre branche avec le code de la master :

```
git checkout <votre-branch>
git rebase master
```

Annuler un rebase en cours :

```
git rebase --abort
```

Continuer un rebase après avoir résolu des conflits :

```
git rebase --continue
```

8. Travailler avec les dépots distants

Lister tous les dépôts distants configurés :

```
git remote -v
```

Monter les informations d'un dépôt distant :

```
git remote show origin
```

Ajouter un nouveau dépôt distant, nommé <remote> :

```
git remote add <remote> <url>
```

Synchroniser la branche origin avec la master en faisant d'origin le dépôt distant par défaut :

```
git push -u origin master
```

Télécharger toutes les modifications d'un dépôt distant nommé <remote>, sans les fusionner :

```
git fetch <remote>
```

Télécharger les modifications et les fusionner directement dans le HEAD :

```
git remote pull <remote> <url>
```

Fusionner les modifications de la master distante sur la branche courante :

```
git pull origin master
```

Récupérer toutes les modifications du HEAD dans le dépôt local :

```
git pull
# ou
git pull origin master
```

Publier les modifications locales sur un dépôt distant :

```
git push
#ou
git push remote <remote> <branch>
```

Publier les tags :

```
git push --tags
```

9. Annulation

Annuler le dernier git add :

```
git reset HEAD
```

Annuler les modifications locales d'un fichier spécifique :

```
git checkout HEAD <file>
```

Annuler un commit (création d'un commit avec les modifications inverses) :

```
git revert <commit>
```

Placer le pointeur du HEAD sur un commit précédent. Conserve toutes les modifications effectuées depuis :

```
git reset <commit>
```

⚠ Annuler toutes les modifications dans le répertoire de travail :

```
git reset --hard HEAD
```

⚠ Placer le pointeur du HEAD sur un commit précédent. Annule toutes les modifications effectuées depuis :

```
git reset --hard <commit>
```