

1. À quoi sert git stash ?

git stash permet de **mettre de côté temporairement** des modifications en cours *sans les committer*.

C'est un **tiroir** où tu ranges momentanément ton travail pour :

- changer de branche
- pull une mise à jour
- tester un truc ailleurs
- revenir plus tard à ton travail en cours

Puis tu peux **récupérer** ces modifications quand tu veux.

2. Exemple simple

Tu modifies 3 fichiers... mais tu dois changer de branche immédiatement.

```
git stash
```

Git enregistre tes modifications **hors vue**, et ton working directory redevient **propre** (clean).

3. Voir la liste des stash

```
git stash list
```

Tu verras des entrées comme :

```
stash@{0}: WIP on main: 87ac123 ajout fonction X
stash@{1}: WIP on feature: ...
```

4. Récupérer un stash

Deux options :

Ramener le stash **et le garder dans la liste** :

```
git stash apply
```

Ramener le stash **et le supprimer automatiquement** :

```
git stash pop
```

pop = apply + delete

apply = juste récupérer

5. Supprimer un stash

```
git stash drop stash@{0}
```

Ou tout effacer :

```
git stash clear
```

6. Stasher seulement certains fichiers

```
git stash push fichier.txt
```

7. Créer un stash avec message

```
git stash push -m "Travail en cours sur la fonction X"
```

8. Stash des fichiers non trackés ?

Par défaut, git ne stash **pas** les fichiers non trackés.

Ajouter aussi les fichiers non trackés :

```
git stash -u
```

Ajouter absolument tout (même ignorés) :

```
git stash -a
```

Résumé visuel

Action	Commande
Ranger temporairement le travail	<code>git stash</code>
Lister les stash	<code>git stash list</code>
Restaurer sans supprimer	<code>git stash apply</code>
Restaurer + supprimer	<code>git stash pop</code>
Stash avec message	<code>git stash push -m "msg"</code>
Inclure non-trackés	<code>git stash -u</code>
Supprimer un stash	<code>git stash drop</code>