Git Cheat Sheet

1. Créer un dépôt

Depuis un dépôt distant

Git clone URL

clone un dépôt Git existant sur un autre serveur.
exemple SSH (utilisation d'une clé SSH) et HTTP (identification par nom d'utilisateur et mot de passe) :

```
git clone git@URL:projet/repo.git
git clone http://URL/projet/repo.git
```

Depuis un dossier local

Git init

prend un projet ou un répertoire existant et l'importe dans Git.

2. Organiser les modifications

Afficher les modifications non appliquées

Git status

- affiche les fichiers modifiés.
- affiche l'état de la branche par rapport au serveur distant.
- indique des suggestions d'aide.

⚠ Ne pas toujours croire git status: la vision locale du serveur distant n'est pas toujours à jour. Il faut actualiser fréquemment avec git fetch.

Ajouter un fichier dans le commit

Git add $fichier

`git add .` ajoute tous les fichiers du dossier en cours.
exemple : `git add $fichier1 $fichier2`

Annuler un git add avant commit

Git reset $fichier

exemple : `git reset $fichier`
Git reset tout court permet de recommencer les adds depuis le dernier commit

Annuler les modifications d'un fichier

Git checkout -- $fichier

⚠ toutes les modifications que vous auriez réalisées sur ce fichier ont **disparu** — vous venez tout juste de l'écraser avec un autre fichier.

Valider les modifications

Git commit -m $message

exemple :

```
git commit -m "Ajout feature dans $fichier1, $fichier2
reste en test"
```

`git commit --amend` permet de modifier le dernier commit.

Remiser son travail

Git stash

prend l'état en cours de votre répertoire de travail et les enregistre dans la pile de stash.
`git stash apply` permet d'appliquer les changements mis dans le stash.
`git stash list` permet de voir le contenu de la pile de stash.

3. Gérer mes branches

Créer une nouvelle branche

Git branch $branche

`git branch -a` permet d'afficher toutes les branches.
`git checkout -b $branche` crée une nouvelle branche et se positionne dessus.

Changer la branche de travail courante

Git checkout $branche

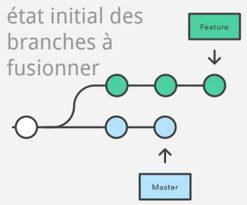
Note : checkout sur une branche distante aura pour effet de la rapatrier, en plus de se placer dessus.
⚠ Dans ce second cas, il faut bien checkout $mabranche et non pas checkout origin/$mabranche, si les deux existent, le deuxième n'a pas l'effet attendu ici.
Exemple : Pour créer une branche de travail bugFix, puis se placer dessus: `git branch bugFix & git checkout bugFix.` Une commande équivalente est: `git checkout -b bugFix`, qui crée la branche bugFix et se positionne dessus.
git checkout -f $branche permet de changer de branche en supprimant les changements locaux. Sans l'option "-f", le checkout n'aurait pas pu être exécuté par Git.

Connaître la branche courante

git branch

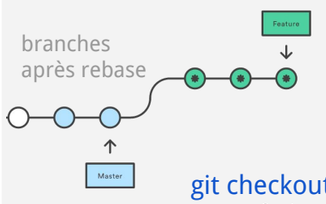
Fusionner des branches

Deux cas principaux:



git rebase $master $feature

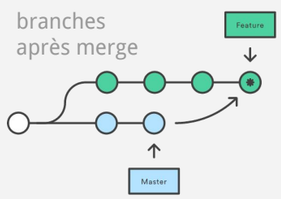
Ici, git rejoue les modifications de feature sur le sommet de master. Feature devient une branche contenant les modifications des deux. Master ne bouge pas. Pour ramener master sur feature on peut utiliser: *git checkout master & git merge feature.*
⚠ git rebase est très pratique pour avoir un historique "propre", car il ne crée pas de commit de fusion. Cependant, **on ne rebase absolument pas les branches qui sont déjà sur le serveur.**



git checkout $feature & git merge $master

fusionne la branche feature avec la branche master:

- Si feature est un ancêtre de master, Git va simplement avancer (fast-forward automatique) feature sur master.
- Sinon, Git les fusionne en créant un nouveau commit de fusion, et avance la branche source sur le nouveau commit comme dans le schéma suivant.



Exemple :

```
git checkout master  // je me place sur la branche master
git merge feature    // je fusionne avec la branche feature
git branch -d feature  // je supprime la branche feature si besoin
```

Supprimer une branche

Lorsque les branches ne sont plus utilisées, il faut les supprimer.
En local: git branch -d branche_à_supprimer
Sur le serveur: git push :branche_à_supprimer

Renommer une branche

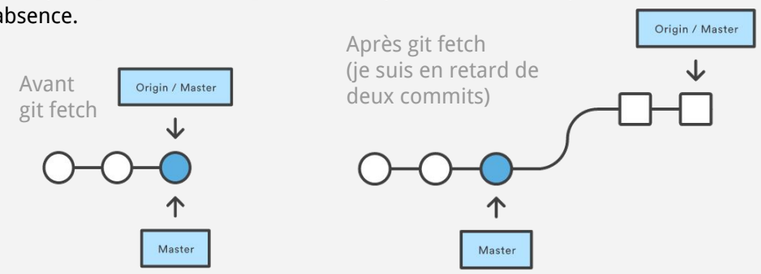
git branch -m $oldname $ newname

4. Échanges avec le serveur

Récupérer les informations sur le serveur

Git fetch

Exemple : `git fetch & git status` permet de voir ce qui a été fait en votre absence.



Nettoyer ma vision locale du serveur

Git fetch --prune

Le fetch prune va supprimer toute les références distantes ne pointant plus vers rien (supprimées du serveur).

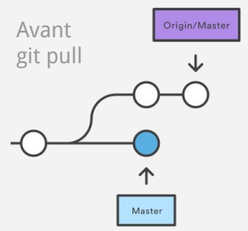
Envoyer vos modifications sur le serveur

Git push $origin $branche

Pour pousser le travail sur le serveur distant une fois que le travail est prêt à être partagé.
L'option *--set-upstream $branche-distante* définit la branche distante cible et peut en créer une. La cible sera utilisée pour les git pull.

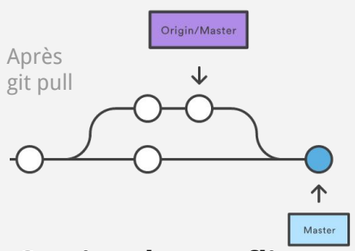
Se mettre à jour avec la branche distante

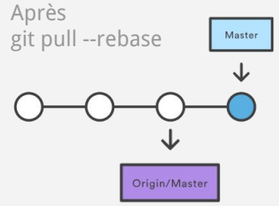
Git pull $origin $branche



Git pull va récupérer les changements connus dans notre branche actuelle. Son comportement est similaire à git merge.

`Git pull --rebase` se comporte comme un rebase.





Gestion des conflits

Git mergetool

Il arrive que la fusion soit impossible à réaliser sans intervention manuelle (conflit). Dans ce cas, le plus simple est d'utiliser la commande git mergetool (pour utiliser git mergetool il faut un outils de fusion comme smartgit ou meld).
Pour **sortir de ce mode** la commande est *git merge --abort* ou *git rebase --abort* en fonction du type de pull.