

Contents

Cheatsheet Git	1
Démarrer et configurer	1
Le quotidien	1
Branches	2
Rebase (historique linéaire)	2
Conflits	2
Tags et versions (SemVer)	3
Annuler (du plus doux au plus brutal)	3
.gitignore	3
Encadré : pièges fréquents	3

Cheatsheet Git

Les 3 zones : **dossier de travail** (vos fichiers) -> git add -> **index** (prochain commit) -> git commit -> **dépôt local** -> git push -> **remote** (GitHub).

Démarrer et configurer

Commande	Effet
git config --global user.name "Nom" / user.email "mail"	Votre identité (une fois par poste)
git init	Transforme le dossier courant en dépôt
git clone git@github.com:user/repo.git	Copie un dépôt distant
git remote -v	Liste les remotes (origin = GitHub)

Le quotidien

Commande	Effet
git status	Où j'en suis ? (à lancer souvent)
git diff/git diff --staged	Changements non indexés / indexés
git add fichier/git add -p	Indexe un fichier / choisit les morceaux
git commit -m "feat: ajoute la liste des tâches"	Enregistre un commit
git log --oneline --graph --all	Historique en arbre
git push/git push -u origin ma-branche	Envoie (le -u mémorise la branche, 1re fois)
git pull/git fetch	Récupère + fusionne / récupère sans fusionner

Message de commit (Conventional Commits) : feat: (nouveau), fix: (correction), docs:, test:, refactor:, chore:, ci:. Impératif, court, précis.

Branches

Commande	Effet
git branch	Liste les branches locales (* = courante)
git switch -c feature/ma-carte	Crée et bascule sur une branche
git switch main	Change de branche
git merge feature/ma-carte	Fusionne cette branche dans la branche courante
git branch -d feature/ma-carte	Supprime une branche déjà fusionnée
git push origin --delete feature/ma-carte	Supprime la branche sur GitHub

Flux **GitHub Flow** : git switch main && git pull > git switch -c feature/x > commits > git push -u origin feature/x > Pull Request > revue > merge > suppression de la branche.

Rebase (historique linéaire)

Commande	Effet
git fetch origin && git rebase origin/main	Rejoue vos commits au-dessus de la dernière main
git pull --rebase	pull sans commit de fusion inutile
git rebase --continue / --abort	Après résolution d'un conflit / pour tout annuler
git push --force-with-lease	Pousse une branche rebasée (variante prudente du « force push »)

Règle d'or : ne jamais rebaser une branche que d'autres utilisent (comme main). Rebaser = réécrire l'historique.

Conflits

Git s'arrête : CONFLICT (content): Merge conflict in index.html.

```
<<<<<< HEAD
<li>Alice</li>          <- votre version (branche courante)
=====
<li>Bob</li>            <- la version qui arrive
>>>>>> feature/carte-bob
```

1. Ouvrez le fichier (VS Code propose *Accept Current / Incoming / Both*). Gardez ce qui est juste, **supprimez les 3 marqueurs** <<<<<<, =====, >>>>>>.
2. Testez que ça marche.
3. git add index.html puis git commit (merge) ou git rebase --continue (rebase).
4. Besoin de tout annuler : git merge --abort / git rebase --abort.

Tags et versions (SemVer)

vMAJEUR.MINEUR.CORRECTIF : correctif sans nouveauté = v1.0.1 ; nouvelle fonction compatible = v1.1.0 ; changement incompatible = v2.0.0.

Commande	Effet
git tag -a v1.0.0 -m "Première version"	Tag annoté sur le commit courant
git tag / git show v1.0.0	Liste / détail
git push origin v1.0.0 (ou git push --tags)	Publie le tag (déclenche les workflows on: push: tags)
git tag -d v1.0.0	Supprime le tag en local

Annuler (du plus doux au plus brutal)

Situation	Commande
Fichier modifié, pas encore add : revenir à la version commitée	git restore fichier
Fichier déjà add : le sortir de l'index	git restore --staged fichier
Corriger le dernier commit (message/oubli), pas encore poussé	git commit --amend
Annuler un commit déjà poussé (sûr : crée un commit inverse)	git revert <sha>
Défaire les 2 derniers commits en gardant les changements	git reset --soft HEAD~2
Détruire tout changement local (irréversible !)	git reset --hard HEAD
Mise de côté temporaire J'ai « perdu » un commit	git stash puis git stash pop git reflog (journal de tout), puis git switch -c rescue <sha>

.gitignore

```
node_modules/  
dist/  
.env  
*.log
```

Un fichier **déjà suivi** reste suivi : git rm --cached fichier puis commit.

Encadré : pièges fréquents

- **Commiter un secret** (.env, mot de passe, jeton) : le supprimer dans un nouveau commit **ne suffit pas** (il reste dans l'historique). Révoquez/changez le secret immédiatement, prévenez le formateur.
- **Travailler sur main** par mégarde : git switch -c ma-branche avant de commiter (les changements suivent).

- Écraser une branche partagée avec un push forcé efface le travail des autres. Si c'est nécessaire (après un rebase de **votre** branche) : `--force-with-lease`, et jamais sur `main`.
- **git reset --hard** et `git clean -fd` détruisent sans corbeille.
- Oublier `git pull` avant de commencer : conflits évitables.
- `HEAD detached` : vous êtes sur un commit, pas une branche ; `git switch -c nom` pour le conserver, ou `git switch main`.
- `git add .` sans `git status` : vous committez des fichiers non voulus (`node_modules`, `.env`).