Cheat Sheet Git - Fork / Clone / Branch / Commit / Push / Pull(Merge) Request + IntelliJ

Page 1 — Faire le travail et ouvrir une PR/MR

1) Workflow recommandé (simple et sûr)

- Fork du repo du cours (sur le site) → tu obtiens un repo à toi (droits d'écriture).
- Clone ton fork sur ton PC.
- Ajoute le repo du prof en upstream (pour récupérer les mises à jour du cours).
- Travaille sur une branche (jamais directement sur main/master).
- Fais des petits commits réguliers, puis push sur ton fork.
- Ouvre une PR/MR vers main du repo du prof et ajoute le prof en reviewer.

2) Commandes essentielles (à copier/coller)

Objectif	Commande
Cloner ton fork	git clone <URL_DE_TON_FORK> cd <nom-du-repo>
Ajouter upstream (une fois)	git remote add upstream <URL_DU_REPO_DU_PROF> git remote -v
Créer une branche	git switch -c feature/td1 # (ou feature/tp2)
Voir ce qui change	git status git diff
Ajouter + commit	git add . git commit -m "TD1: ajout des tests sur ..."
Push (1ère fois)	git push -u origin feature/td1
Push (ensuite)	git push

3) Ouvrir une Pull Request / Merge Request (PR/MR)

- Sur le site (GitHub/GitLab) : bouton Create Pull Request / Create Merge Request.
- Source : ta branche sur ton fork — Cible : main du repo du prof.
- Ajoute le prof en reviewer (obligatoire).
- Dans la description : ce qui est fait, ce qui reste, comment tester.

Checklist avant d'ouvrir la PR/MR

- Les tests passent en local (commande : mvn test ou ./gradlew test).
- Le pipeline CI est vert (si activé).
- Tu n'as pas commité des dossiers inutiles (ex : target/, .idea/).
- Commits lisibles et messages explicites (pas "update").

Page 2 — Mises à jour du cours, conflits et IDE (IntelliJ / Eclipse)

4) Récupérer les mises à jour du cours (upstream)

But : récupérer les derniers supports/corrections du repo du prof, sans perdre ton travail.

Étape	Commande
Mettre à jour main	git switch main git fetch upstream git merge upstream/main git push origin main
Mettre à jour ta branche	git switch feature/td1 git merge main # (option avancée : git rebase main)

5) Conflits : procédure minimale

- Git annonce des conflits → fais : git status pour voir les fichiers concernés.
- Ouvre les fichiers en conflit et garde la bonne version (ou combine).
- Valide la résolution : git add <fichier>.
- Si tu étais en merge : git commit ; si tu étais en rebase : git rebase --continue.
- Termine par : git push.
- Débutant : évite git push --force.

6) IntelliJ IDEA : faire la même chose sans terminal

- Cloner : écran d’accueil → Get from VCS → URL de ton fork → Clone.
- Créer une branche : en bas à droite (nom de branche) → New Branch… → feature/td1.
- Voir les changements : fenêtre Git → Local Changes → diff.
- Commit : Commit (Ctrl+K) → message → cocher fichiers → Commit.
- Push : Push (Ctrl+Shift+K) → vérifier la branche → Push.
- Pull / Fetch : menu Git → Fetch ou Pull….
- Conflits : IntelliJ ouvre un outil de fusion → résoudre → Apply → commit.

7) Eclipse : faire la même chose avec EGit

- Cloner : File → Import… → Git → Projects from Git → Clone URI → URL de ton fork → Finish.
- Importer dans le workspace : choisir Import existing Eclipse projects (ou General project si nécessaire).
- Créer une branche : clic droit projet → Team → Switch To → New Branch… → feature/td1.
- Voir les changements : Git Staging view → fichiers modifiés + diff.
- Commit : dans Git Staging → glisser vers Staged Changes → message → Commit.
- Push : clic droit projet → Team → Push Branch… (ou bouton Push dans Git Staging).
- Fetch/Pull : clic droit projet → Team → Fetch from Upstream / Pull.
- Conflits : EGit propose un merge tool → résoudre → marquer comme résolu → commit.

8) Problèmes fréquents (solutions rapides)

- fatal: not a git repository → tu n’es pas dans le bon dossier (cd).
- nothing to commit → tu n’as pas de changements ou tu n’as pas fait git add.

- rejected (non-fast-forward) au push → fais git pull --rebase puis git push.
- Tu as commité sur main par erreur → crée une branche et continue dessus (demande aide si besoin).