

TD 2a — Un premier dépôt

Guide détaillé, étape par étape

Le TD crée le dépôt git du projet de la séance, une petite calculatrice en Python, et y enregistre un premier fichier. Il correspond aux questions 1 à 5 de la feuille, et dure une quinzaine de minutes. Les TD 2b, 2c, 2d et 2e reprennent ce même dépôt : il ne faut pas le supprimer à la fin.

Le TD se fait dans un terminal bash, avec git. Sur les postes de la salle, c'est la fenêtre Git Bash ; la page Git et Git Bash décrit comment l'ouvrir.

Étape	Objectif	Questions	Ce qu'on fait
1	afficher l'historique d'un dépôt sous forme de graphe	1	enregistrer l'alias <code>git llog</code> , qui affiche le graphe du projet
2	créer un dépôt git	2 et 3	créer le dossier du projet, et y initialiser un dépôt
3	enregistrer un premier état du projet	4 et 5	créer <code>README.md</code> , puis l'enregistrer en un premier commit

Chaque étape commence par un encadré qui la résume. Ce que chaque étape fait constater est expliqué à la fin du guide, dans « Ce que le TD fait constater » : faire l'étape d'abord, et noter ce qu'on observe, avant de lire l'explication.

1 · Enregistrer l'alias `git llog`

À faire : ouvrir Git Bash ; vérifier que git répond ; enregistrer l'alias `llog`.

À obtenir : `git llog` est une commande reconnue.

Ouvrir le terminal

Ouvrir Git Bash comme l'indique la page Git et Git Bash. L'invite a la forme suivante, sur deux lignes :

```
eleve@POSTE MINGW64 ~  
$
```

Taper `git --version`, puis Entrée.

Vérification : le terminal affiche `git version 2.` suivi d'un numéro.

L'alias

Taper la commande suivante sur une seule ligne, telle quelle, apostrophes comprises :

```
git config --global alias.llog 'log --graph --pretty=oneline --abbrev-commit --decorate'
```

La commande n'affiche rien. Elle enregistre dans la configuration de git un nouveau nom, `llog`, pour la commande entre apostrophes. Taper ensuite `git llog` revient à taper `git log` suivi des quatre options. L'option `--global` enregistre l'alias pour tous les dépôts du compte, et non pour un seul.

Vérification : `git config --global alias.llog` affiche `log --graph --pretty=oneline --abbrev-commit --decorate`.

2 · Créer le dossier du projet, et le dépôt

À faire : se placer dans travail/ du dossier du TD ; y créer projet_2 ; entrer dedans ; y lancer `git init` ; régler le nom et l'adresse de l'auteur des commits.

À obtenir : l'invite se termine par projet_2 (master).

Le dossier du TD

Le dossier du TD est dans l'archive de la séance, extraite dans C:\Users\eleve\Desktop\info01\ :

```
2a_premier_depot\  
├─ travail\           vide : le projet s'y crée  
├─ README.md         la présentation du TD  
└─ td_2a_premier_depot.pdf la feuille du TD
```

Dans Git Bash, ce dossier s'écrit ~/Desktop/info01/cours2/2a_premier_depot. La touche Tab complète les noms : taper `cd ~/De`, puis Tab, puis la suite.

```
cd ~/Desktop/info01/cours2/2a_premier_depot/travail  
pwd
```

Vérification : `pwd` affiche /c/Users/eleve/Desktop/info01/cours2/2a_premier_depot/travail.

Le dossier du projet et le dépôt

```
mkdir projet_2  
cd projet_2  
git init
```

`mkdir` et `cd` n'affichent rien. `git init` affiche une ligne qui donne le chemin du dépôt créé, et qui se termine par projet_2/.git/ : en français, Dépôt Git vide initialisé dans ../projet_2/.git/ ; en anglais, Initialized empty Git repository in ../projet_2/.git/.

Selon la configuration du poste, `git init` affiche avant cette ligne un paragraphe commençant par astuce:, sur le nom de la branche initiale. Il n'appelle aucune action.

Vérification : l'invite se termine maintenant par projet_2 (master). Git Bash affiche entre parenthèses la branche courante quand le dossier courant est un dépôt. Si l'invite affiche (main), le poste est réglé pour nommer ainsi la première branche : lire main partout où les TD écrivent master.

Le nom de l'auteur des commits

Chaque commit porte le nom et l'adresse de son auteur. Sur un poste de la salle, le compte eleve est commun à tous : régler son nom dans le dépôt, et non pour tout le compte.

```
git config user.name "Prénom Nom"  
git config user.email "prenom.nom@example.fr"
```

Remplacer les valeurs entre guillemets par les siennes. Sans ce réglage, et si aucun nom n'est enregistré pour le compte, `git commit` s'arrête à l'étape suivante sur le message Identité d'auteur inconnue, suivi de Veuillez me dire qui vous êtes. (en anglais, Author identity unknown et Please tell me who you are.).

3 · Enregistrer un premier commit

À faire : créer README.md dans projet_2 et y écrire une ligne ; l'ajouter avec `git add` ; l'enregistrer avec `git commit` ; lire `git status` et `git llog` entre chaque commande.

À obtenir : `git llog` affiche un commit, ajout du README.

Créer le fichier

Le fichier est demandé en Markdown, le format de documentation vu au cours 1 : il prend l'extension .md. Le plus simple est de l'écrire depuis le terminal :

```
echo "# Geo Calculatrice" > README.md
```

`echo` affiche le texte qui le suit, et `>` envoie cet affichage dans le fichier README.md, qu'il crée. Le fichier peut aussi être créé dans VS Code, dossier projet_2 ouvert.

```
git status
```

À noter : dans quelle rubrique `git status` liste README.md.

L'ajouter, puis l'enregistrer

```
git add README.md
git status
```

À noter : dans quelle rubrique README.md apparaît maintenant.

```
git commit -m "ajout du README"
```

`-m` donne le message du commit sur la ligne de commande. Git affiche le commit créé :

```
[master (commit racine) 1bab660] ajout du README
1 file changed, 1 insertion(+)
create mode 100644 README.md
```

L'identifiant, ici 1bab660, est différent sur chaque poste.

```
git status
git llog
```

Vérification : `git status` affiche rien à valider, la copie de travail est propre, et `git llog` affiche une ligne :

```
* 1bab660 (HEAD -> master) ajout du README
```

Ce que le TD fait constater

Cette section se lit après avoir fait les étapes.

Le dépôt est le dossier `.git`

`git init` a créé un seul dossier, `.git`, caché, dans `projet_2`. `ls -a` l'affiche. Tout ce que git enregistre est dans ce dossier ; les fichiers de `projet_2` restent des fichiers ordinaires. Si `git init` a été lancé au mauvais endroit, par exemple dans `travail/` au lieu de `projet_2`, supprimer le dossier `.git` créé là (`rm -rf .git`, depuis ce dossier) défait l'opération. C'est l'erreur la plus fréquente du TD : `pwd` avant `git init` l'évite.

Les trois états de `README.md`

Après	<code>git status</code> range <code>README.md</code> sous	L'état du fichier
sa création	« Fichiers non suivis »	git le voit, mais ne l'enregistre pas
<code>git add</code>	« Modifications qui seront validées », nouveau fichier	il est dans la zone de préparation
<code>git commit</code>	rien : « la copie de travail est propre »	il est enregistré dans un commit

`git add` ne crée pas de commit : il prépare ce que le prochain commit contiendra. Un commit peut ainsi réunir plusieurs fichiers ajoutés un par un, et laisser de côté ceux qu'on n'a pas encore ajoutés.

Le premier commit

Git a marqué ce commit (commit racine) : c'est le seul du projet qui n'a pas de parent. `git log` l'affiche avec deux noms : `master`, la branche créée par `git init`, et `HEAD`, qui désigne le commit sur lequel on se trouve. La flèche `HEAD -> master` indique que `HEAD` suit la branche `master` : le prochain commit fera avancer `master`. La partie « Branches, fusion et conflits » détaille ces deux noms.