

TD 2e — Publier une version

Guide détaillé, étape par étape

Le TD termine le projet de la séance : il fusionne develop dans master, puis pose sur le commit obtenu le tag v1.0, qui nomme la première version de la calculatrice. Il correspond aux questions 26 et 27 de la feuille, et dure une dizaine de minutes.

On travaille dans le dépôt du TD 2a, tel que le TD 2d l'a laissé, ~/Desktop/info01/cours2/2a_premier_depot/travail/projet_2, dans Git Bash. Le TD n'a pas de fichier de départ.

Étape	Objectif	Questions	Ce qu'on fait
1	reporter le travail terminé sur la branche principale	26	fusionner develop dans master
2	nommer une version livrée	27	poser le tag v1.0, et vérifier la version

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 · Fusionner develop dans master

À faire : vérifier que la fusion du TD 2d est terminée ; se placer sur master ; y fusionner develop.

À obtenir : master et develop désignent le même commit.

```
cd ~/Desktop/info01/cours2/2a_premier_depot/travail/projet_2
git status
```

Vérification : `git status` n'affiche ni « chemins non fusionnés » ni « la fusion n'est pas terminée ». Sinon, terminer d'abord le TD 2d.

```
git checkout master
git merge develop
```

```
Mise à jour 1bab660..4ced033
Fast-forward
 README.md      |  4 ++++
src/main.py     | 42 +++++
src/operations.py | 14 +++++
3 files changed, 60 insertions(+)
create mode 100644 src/main.py
create mode 100644 src/operations.py
```

À noter : le mot que `git merge` affiche à sa deuxième ligne.

2 · Poser le tag

À faire : poser le tag v1.0 sur le commit courant ; afficher les tags et le graphe ; lancer le programme.

À obtenir : git llog affiche tag: v1.0 sur le commit de master.

```
git tag -a v1.0 -m "première version de la calculatrice"
git tag
git llog
```

-a crée un tag annoté, qui enregistre son auteur, sa date et un message, donné par -m. git tag liste les tags du dépôt : v1.0.

Vérification : la première ligne du graphe porte trois noms :

```
* 4ced033 (HEAD -> master, tag: v1.0, develop) Merge branch 'main_code' into develop
```

```
python src/main.py
```

Le programme affiche son titre et calcule l'opération tapée : la version taguée est une version qui fonctionne.

Ce que le TD fait constater

Cette section se lit après avoir fait les étapes.

Une avance rapide

La fusion affiche Fast-forward. `master` n'a reçu aucun commit depuis le TD 2a : tout le travail s'est fait sur `develop` et sur les branches de tâche. Il n'y avait rien à réunir, et `git` a déplacé `master` jusqu'au dernier commit de `develop`. Le résumé liste tous les fichiers apparus depuis le premier commit, parce que `master` passe d'un coup du TD 2a à la fin du TD 2d.

Le tag nomme une version

`v1.0` désigne le commit `4ced033`, sur lequel se trouvent `master` et `develop`. Les deux branches avanceront avec les prochains commits ; le tag, lui, reste sur ce commit. `git checkout v1.0` redonnera à tout moment les fichiers de la première version, sans chercher son identifiant. `git show v1.0` affiche le tag, son auteur, sa date et son message, puis le commit qu'il désigne.

L'organisation du projet

Le graphe final suit l'organisation décrite dans « Lire et tenir un dépôt » : `master` ne porte que le premier commit et la version terminée, taguée ; `develop` a réuni le travail ; chaque tâche a eu sa branche. La version suivante se préparerait de la même façon : de nouvelles branches depuis `develop`, fusionnées dans `develop`, puis `develop` fusionnée dans `master`, et un tag `v1.1` ou `v2.0`.