Git et GitHub

Versionner et collaborer sur son code

BUT Science des Données

Université de Poitiers

2025-2026

Plan

Sans gestion de versions

Ce qui arrive souvent :

- `projet_v1.py`
- `projet_v2.py`
- `projet_final.py`
- `projet_final_v2.py`
- `projet_VRAI_final.py`

Problèmes :

- Impossible de revenir en arrière proprement
- Difficile de collaborer à plusieurs
- Pas de traçabilité des modifications
- Risque de perte de travail

Git : définition

Qu'est-ce que Git ?

Système de **contrôle de version distribué** créé par Linus Torvalds (2005). Il permet de suivre l'historique des modifications d'un projet.

Avantages :

- Historique complet de chaque modification
- Travail en parallèle avec les **branches**
- Collaboration facilitée via des plateformes (GitHub)
- Retour en arrière possible à tout moment

Les trois zones de Git

Working Directory Vos fichiers de travail

Staging Area Zone de préparation du prochain commit

Repository Historique des commits (dossier `.git/`)

Qu'est-ce qu'un commit ?

Commit = instantané du projet

Un commit enregistre l'état de tous les fichiers suivis à un instant donné.

Chaque commit contient :

- Un **identifiant unique** (hash SHA-1)
- L'**auteur** et la **date**
- Un **message** décrivant les modifications
- Un **pointeur** vers le commit parent

Installer et configurer Git

Installation :

```
# Linux (Ubuntu/Debian)
sudo apt install git

# macOS
brew install git

# Windows : télécharger depuis git-scm.com
```

Configuration initiale :

```
git config --global user.name "Votre Nom"
git config --global user.email "votre@email.fr"
```

Initialiser et suivre

```
# Créer un nouveau dépôt Git
git init

# Voir l'état des fichiers
git status

# Ajouter un fichier à la staging area
git add mon_fichier.py

# Ajouter tous les fichiers modifiés
git add .
```

Commiter et consulter l'historique

```
# Créer un commit
git commit -m "Ajouter le script principal"

# Voir l'historique des commits
git log

# Historique compact
git log --oneline

# Voir les différences non commitées
git diff
```

GitHub

Qu'est-ce que GitHub ?

Plateforme d'hébergement de dépôts Git en ligne. Permet la collaboration, le partage et la sauvegarde du code.

Fonctionnalités clés :

- Hébergement gratuit de dépôts publics et privés
- Pull Requests pour la revue de code
- Issues pour le suivi des bugs
- GitHub Actions pour l'automatisation (CI/CD)

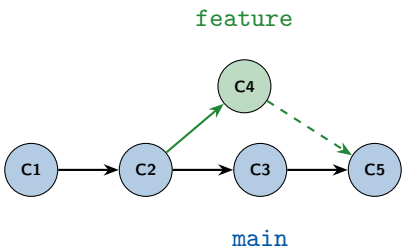
Lier un dépôt local à GitHub

```
# Ajouter un dépôt distant
git remote add origin https://github.com/USER/REPO.git

# Envoyer les commits vers GitHub
git push -u origin main

# Récupérer les modifications depuis GitHub
git pull origin main

# Cloner un dépôt existant
git clone https://github.com/USER/REPO.git
```

Concept de branche

Branche = ligne de développement indépendante

Permet de travailler sur une fonctionnalité sans affecter le code principal.



Commandes de branches

```
# Créer une branche
git branch ma-feature

# Basculer sur la branche
git switch ma-feature

# Créer et basculer en une commande
git switch -c ma-feature

# Lister les branches
git branch

# Fusionner une branche dans main
git switch main
git merge ma-feature
```

Qu'est-ce qu'un conflit ?

Conflit

Se produit quand deux branches modifient la **même ligne** du même fichier. Git ne peut pas fusionner automatiquement.

Marqueurs de conflit dans le fichier :

```
<<<<<<< HEAD
resultat = a + b
=======
resultat = a * b
>>>>>>> ma-feature
```

Résolution :

1. Ouvrir le fichier et choisir la bonne version
2. Supprimer les marqueurs «««<, =======, »»»>
3. `git add` puis `git commit`

.gitignore

Fichier `.gitignore` : liste les fichiers à ne **pas** suivre.

```
# Fichiers Python
__pycache__/
*.pyc
.env

# Données et sorties
output/
data/raw/

# IDE
.vscode/
.idea/
```

Toujours créer un `.gitignore` au début du projet.

Messages de commit

Un bon message de commit

- Court et descriptif (max 72 caractères)
- Commence par un verbe à l'impératif
- Décrit **ce qui a changé** et **pourquoi**

Bons exemples :

✓ `Ajouter la validation des emails`

✓ `Corriger le bug de connexion MySQL`

Mauvais exemples :

× `fix`

× `modifications diverses`

× `ça marche enfin`

Ce qu'il faut retenir

Git

- Système de contrôle de version distribué
- Trois zones : working dir, staging area, repository
- Commandes : `init`, `add`, `commit`, `log`, `diff`

GitHub et collaboration

- Dépôt distant : `push`, `pull`, `clone`
- Branches pour le travail en parallèle
- Résolution de conflits lors du merge

Bonnes pratiques

- Toujours un `.gitignore`
- Messages de commit clairs et descriptifs