## TP : Manipulation de GitHub et GitLab

**Durée : 3h**  
**Objectifs :**

* Comprendre et utiliser Git en local
* Travailler avec GitHub et GitLab
* Gérer un projet avec Git (clonage, commit, push, pull, merge)
* Collaborer via des pull requests et merge requests
* Résoudre des conflits

**Prérequis**

* Avoir installé **Git** (git --version pour vérifier)
* Avoir un compte **GitHub** et un compte **GitLab**
* Avoir un éditeur de code installé (par exemple **VS Code**)

**Partie 1 : Initialisation et configuration de Git**

**1.1 Configuration de Git**

1. Ouvrez un terminal et configurez votre identité :

|  |
| --- |
| git config --global user.name "VotreNom"  git config --global user.email "VotreEmail@example.com" |

1. Vérifiez la configuration :

|  |
| --- |
| git config --list |

**1.2 Initialisation d’un dépôt local**

1. Créez un dossier pour votre projet et placez-vous dedans :

|  |
| --- |
| mkdir projet-git  cd projet-git |

1. Initialisez Git dans ce dossier :

|  |
| --- |
| git init |

1. Vérifiez que le dépôt est bien initialisé :

|  |
| --- |
| ls -a *# Vérifier la présence du dossier .git* |

**Partie 2 : Utilisation de Git en local**

**2.1 Création et gestion des commits**

1. Créez un fichier README.md et ajoutez du contenu dedans.
2. Ajoutez ce fichier à l’index :

|  |
| --- |
| git add README.md |

1. Effectuez un commit :

|  |
| --- |
| git commit -m "Ajout du fichier README" |

1. Vérifiez l’historique des commits :

|  |
| --- |
| git log --oneline |

**2.2 Création et changement de branches**

1. Créez une nouvelle branche master :

|  |
| --- |
| git branch master |

1. Changez de branche :

|  |
| --- |
| git checkout master |

1. Vérifiez les branches existantes :

|  |
| --- |
| git branch |

**Partie 3 : GitHub et GitLab - Hébergement et collaboration**

**3.1 Hébergement sur GitHub**

1. Connectez-vous à [GitHub](https://github.com) et créez un dépôt **projet-git**.
2. Liez votre dépôt local à GitHub :

|  |
| --- |
| git remote add origin https://github.com/VotreUtilisateur/projet-git.git |

1. Envoyez votre branche main sur GitHub :

|  |
| --- |
| git push -u origin master |

1. Créez un nouveau fichier index.html dans votre projet :

|  |
| --- |
| touch index.html |

1. Ouvrez le fichier index.html dans un éditeur de texte et ajoutez le contenu suivant :

|  |
| --- |
| <!DOCTYPE html>  <html lang="fr">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Accueil</title>  </head>  <body>  <h1>Bonjour</h1>  </body>  </html> |

1. Ajoutez le fichier à Git :

|  |
| --- |
| git add index.html |

1. Validez (commit) le fichier avec un message explicite :

|  |
| --- |
| git commit -m "Ajout du fichier index.html avec Bonjour" |

1. Envoyez (push) les modifications sur GitHub :

|  |
| --- |
| git push origin master |

**3.2 Collaboration avec GitHub (Pull Request)**

1. Sur GitHub, créez une branche feature-1 via l'interface ou en ligne de commande :

|  |
| --- |
| git checkout -b feature-1 |

1. Modifiez le fichier index.html (changer le message bonjour) et effectuez un commit :

|  |
| --- |
| git commit -m "Ajout d’une modification sur feature-1" |

1. Poussez la branche sur GitHub :

|  |
| --- |
| git push -u origin feature-1 |

1. Sur GitHub, créez une Pull Request depuis feature-1 vers main, puis fusionnez-la.
2. Récupérez les modifications en local :

|  |
| --- |
| git checkout master  git pull origin master |

**3.3 Hébergement sur GitLab**

1. Connectez-vous à [GitLab](https://gitlab.com) et créez un dépôt **projet-gitlab**.
2. Sortez du dossier actuel, créez un nouveau dossier, puis exécutez la commande :

|  |
| --- |
| cd new\_project |

1. Appliquer les numéros 4, 5, 6, de la partie 3.1
2. Liez votre dépôt local à GitLab :

|  |
| --- |
| git remote add gitlab https://gitlab.com/VotreUtilisateur/projet-gitlab.git |

1. Poussez le projet sur GitLab :

|  |
| --- |
| git push -u gitlab master |

**3.4 Collaboration avec GitLab (Merge Request)**

1. Créez une branche feature-2 :

|  |
| --- |
| git checkout -b feature-2 |

1. Modifiez le fichier index.html (changer le message bonjour) et committez :

|  |
| --- |
| git commit -m "Ajout d’une modification pour GitLab" |

1. Poussez la branche sur GitLab :

|  |
| --- |
| git push -u gitlab feature-2 |

1. Sur GitLab, créez une Merge Request et fusionnez-la dans main.
2. Récupérez les modifications en local :

|  |
| --- |
| git checkout main  git pull gitlab main |

**Partie 4 : Gestion des conflits et bonnes pratiques**

**4.1 Simulation d’un conflit Git**

1. Sur la branche main, modifiez README.md et committez.
2. Sur une branche conflict-test, modifiez la même ligne de README.md et committez.
3. Essayez de fusionner conflict-test dans main :

|  |
| --- |
| git checkout main  git merge conflict-test |

1. Résolvez le conflit dans l’éditeur, puis terminez la fusion :

|  |
| --- |
| git add README.md  git commit -m "Résolution du conflit" |

**4.2 Bonnes pratiques Git**

* **Nommer clairement les commits** (git commit -m "Correction du bug X")
* **Utiliser les branches de fonctionnalités** (feature-X, fix-Y)
* **Effectuer des pulls réguliers** (git pull)
* **Revoir le code avant d’accepter une Pull/Merge Request**

**Partie 5 : Création et Gestion des Releases sur GitHub et GitLab**

Dans cette section, nous allons apprendre à **créer une release (version)** sur **GitHub** et **GitLab** en suivant les bonnes pratiques.

### 5.1 ****Créer une Release sur GitHub****

**Accéder à ton dépôt :**

Va sur [GitHub](https://github.com) et connecte-toi.

Va dans le dépôt dans lequel tu souhaites créer une release.

**Aller à l'onglet "Releases" :**

Clique sur l'onglet **"Releases"** dans le menu supérieur du dépôt (juste au-dessus des fichiers de ton projet).

**Créer une nouvelle Release :**

Clique sur **"Draft a new release"** sur la droite de la page.

**Détails de la Release :**

**Tag version :** Indique le tag de la version (par exemple v1.0.0). Si tu n'as pas encore de tag pour cette version, tu peux en créer un ici.

**Target Branch :** Sélectionne la branche cible (par exemple, main ou master).

**Release Title :** Donne un titre à ta release (par exemple, "Version 1.0.0").

**Description :** Ajoute une description des modifications de cette version (par exemple, corrections de bugs, nouvelles fonctionnalités, etc.).

Tu peux aussi **télécharger des fichiers** (binaire, .zip, etc.) si nécessaire, en les glissant dans la section **"Attach binaries by dropping them here"**.

**Publier la Release :**

Une fois tous les champs remplis, clique sur **"Publish release"** pour rendre la version disponible.

### 5.2. Créer et gérer des Releases sur GitLab

Créer une Release sur GitLab

Accéder à ton dépôt :

**Va sur** [**GitLab**](https://gitlab.com) **et connecte-toi.**

**Ouvre ton projet dans lequel tu veux créer une release.**

Accéder à l'onglet "Releases" :

**Dans le menu à gauche, va dans** "Repository" **puis clique sur** "Releases" **sous cette section.**

Créer une nouvelle Release :

**Clique sur** "New release" **en haut à droite de la page.**

Détails de la Release :

Tag Name : **Saisis le nom du tag (par exemple v1.0.0). Si le tag n'existe pas encore, GitLab va le créer pour toi.**

Release Title : **Donne un titre à ta release.**

Description : **Ajoute une description de cette version (notes de mise à jour, nouvelles fonctionnalités, corrections, etc.).**

Attacher des fichiers : **Comme sur GitHub, tu peux télécharger des fichiers (binaires, archives, etc.) en les glissant dans la section** "Assets"**.**

Publier la Release : **Une fois tous les champs remplis, clique sur** "Create release" **pour publier la version.**