# TD Git et GitHub

*Git est une application installée (en local) qui permet de gérer les versions d’un logiciel dans son développement.*

*GitHub est une application distante (web) qui permet de synchroniser des développements gérés par Git, ce qui permet de réaliser un développement logiciel en mode collaboratif.*

## Préparation

### Créez et paramétrez votre compte GitHub

1. créez un compte (<https://github.com/>)
2. créez un jeton d’authentification (*token*), qui vous permettra de vous connecter sur votre compte GitHub par https
   * Votre logo > Settings > Developper settings > Personnal access tokens > Tokens (classic) > Generate new token (classic)
   * **tout sélectionner**
   * copier le mot-de-passe dans un document

### Créez votre dépôt CNUM

1. créez un dépot (**un seul dépôt par groupe projet CNUM**)
   * Create a new repository
2. ajoutez les membres du projet comme collaborateurs ainsi que votre enseignant
   * Settings > Collaborators > add people
3. réalisez l’étape A (édition du document README.md sur GitHub)
   * vous pouvez enrichir le document en utilisant la syntaxe Markdown : https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet

### Installez, lancez et parametrez Git

1. installez Git sur votre ordinateur personnel (https://git-scm.com/)
   * connectez vous en tant qu’administrateur pour réaliser l’installation
     + *sinon votre antivirus pourrait bloquer les connexions de Git vers GitHub en les interprétant comme malveillantes*
   * acceptez les nombreux choix par défaut qui vous sont proposés lors de l’installation
2. lancez la console Git (*Git bash*)
3. placez-vous dans le répertoire (dossier) de votre choix
   * pour obtenir le nom du répertoire courant : pwd
   * pour obtenir la liste des fichiers présents dans le répertoire courant : ls
   * pour changer de répertoire : cd [nom\_du\_dossier]
4. munissez vous de la fiche mémo github-git-cheat-sheet\_FR.pdf disponible sur Moodle
   * *les commandes git sont indiquées dans la suite du TD, mais il faudra vous référer à cette fiche pour préciser les arguments (indiqués par ... dans le sujet)*
5. paramétrez Git
   * indiquez votre pseudo GitHub et votre email
     + git config [...]
   * précisez que vous souhaitez la fusion des documents en cas de conflit (utile plus loin)
     + git config --global pull.rebase false

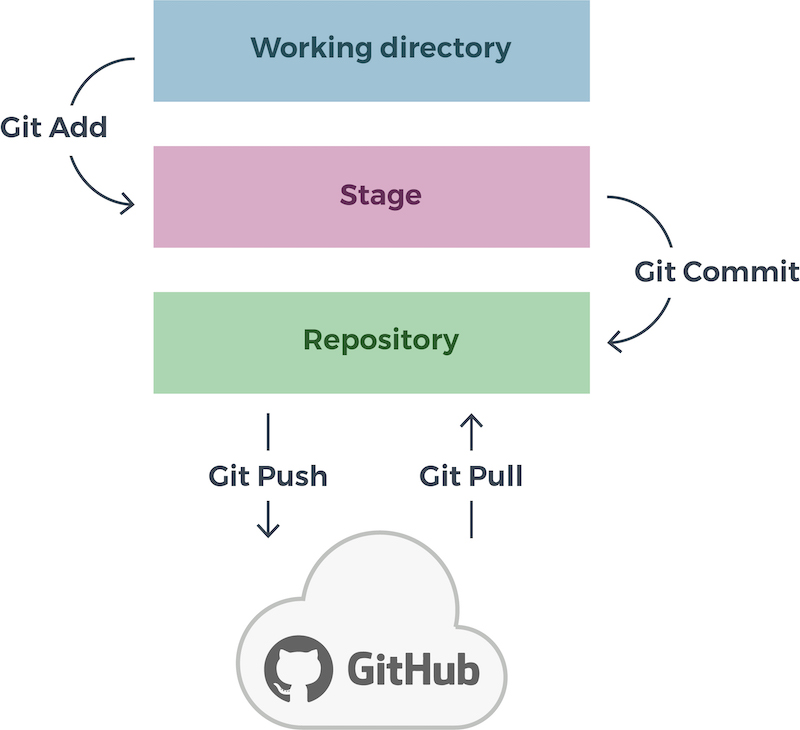
## Développement collaboratif

### Téléchargez votre projet

1. copiez l’adresse du dépot du projet sur GitHub (https://github.com/votre\_pseudo/votre\_projet)
2. téléversez votre projet sur votre ordinateur
   * git clone [...]
   * si demandé (cas d’un projet privé), indiquez votre pseudo GitHub et utilisez le jeton que vous avez créé comme mot-de-passe
   * placez vous dans le répertoire du projet et visualisez son contenu
     + *vous avez bien un fichier README.md sur votre ordinateur :)*

### Développez en collaboratif - partie 1 (documents séparés dans la branche principale)

1. lancez RStudio
   * vérifiez que vous utilisez tous **le même encodage UTF8**, changez pour UTF8 si besoin
     + Tools > Global options > Code > Saving > Serialization > Default text encoding > UTF8
   * placez vous dans le répertoire du projet
   * sélectionnez ce répertoire comme répertoire de travail (*working directory*) sur R
2. développez dans un document séparé
   * réalisez l’étape B (création d’un script R; **chaque membre du groupe travaille sur un script différent**)
3. créez une version en local
   * ajouter le script R à l’index
     + git add [nom\_du\_fichier]
   * créez une nouvelle version
     + git commit [...]
4. téléversez votre script sur GitHub
   * git push origin main
     + donnez votre identifiant GitHub
     + donnez le jeton créé précédemment
   * *rafraichissez la page GitHub pour observer la mise à jour :)*
5. récupérez le travail de vos collaborateurs sur votre ordinateur
   * git pull



Working/Stage/Commit

### Développez en collaboratif - partie 2 (documents communs dans la branche principale)

1. réalisez l’étape C (modification simultanée du document README.md)
2. ajoutez README.md à l’index, créez une version et à tour de rôle téléversez le script sur GitHub
3. résolvez les conflit qui apparaissent, pour celà, si conflit
   * récupérez la dernière version du projet
   * éditez README.md pour régler le conflit
   * repassez à l’étape 2
4. *rafraichissez la page GitHub pour observer la mise à jour :)*
5. identifiez qui a modifié quoi et quand
   * git blame [nom\_du\_fichier]
     + *Git garde les traces précises de toutes les modifications :)*

### Développez en local - partie 1 (gestion des versions)

1. **récupérez la dernière version du projet**
2. identifiez les différences entre les niveaux travail/indexé/version (*working/stage/commit*)
   * réalisez l’étape D (modification de votre script R)
   * identifiez les différences
     + *working VS commit*
       - git status
     + *working VS stage*
       - git diff
     + *stage VS commit*
       - git diff --staged
   * indexez le document **mais ne créez pas de version**
   * réalisez l’étape E (modification de votre script R)
   * identifiez de nouveau les différences *working VS commit*, *working VS stage* et *stage VS commit*
3. revenez à une version antérieure (pas plus ancienne que le dernier commit)
   * annulez les modifications courantes en les remplaçant par la version indexée (*stage -> working*)
     + git restore [nom\_du\_fichier]
     + *vous repassez ainsi à la fin de l’étape D*
   * réalisez l’étape F (modification de votre script R)
   * supprimez votre script de l’index (*commit -> stage*)
     + git restore --staged [nom\_du\_fichier]
   * rétablissez le contenu courant depuis le dernier commit
     + *commit -> working*
       - git restore --worktree [nom\_du\_fichier]
       - *vous repassez ainsi à la fin de l’étape C*
     + (Rq : il est aussi possible de revenir à la dernière version dans l’espace de travail et l’index; *commit -> working & stage*; git restore --worktree --staged [nom\_du\_fichier])
4. visionner le contenu d’une version antérieure (plus ancienne que le dernier commit)
   * obtenez l’historique des versions
     + git log
   * revenez à une version antérieure
     + git checkout [...]
   * observez le contenu du dossier du projet
     + *vous voyagez dans le temps :)*
   * **revenez à la version actuelle**
     + git checkout main
5. supprimez la dernière version en local
   * effectuez l’étape G
   * indexez les modifications
   * supprimez le fichier README.md
     + git rm [nom\_du\_fichier]
   * créez une nouvelle version du projet
   * obtenez l’historique des versions
   * supprimez le dernier commit et revenez au commit précédent (**attention cette opération ne peut être annulée**)
     + git reset --hard HEAD^
   * obtenez l’historique des versions et visualisez le contenu du projet

### Développez en local - partie 2 (branches annexes)

1. **récupérez la dernière version du projet**
2. démarrez une nouvelle branche
   * créez une nouvelle branche
     + git branch [...]
   * vérifiez dans quelle branche vous êtes
     + git branch
   * positionnez vous dans la nouvelle branche
     + git checkout [...]
3. développez dans une branche
   * **vérifiez que vous êtes bien dans la branche annexe**
   * effectuez l’étape H
   * créez une nouvelle version dans la branche
4. visualiser le contenu d’autres branches
   * retournez dans la branche principale
     + git checkout main
   * observez le contenu du projet
   * **retournez dans votre branche annexe**
     + git checkout [...]
   * observez le contenu du projet
     + *vous voyagez dans des mondes parallèles :)*
5. fusionnez la branche annexe à la branche principale
   * retournez dans branche principale
   * vérifiez que vous êtes actuellement dans la branche principale
   * fusionnez la branche annexe à la branche principale
     + git merge [...]
6. Mettez à jour les documents partagés
   * téléversez votre travail sur GitHub
     + *il ne devrait pas y avoir de conflit car vous travaillez sur des lignes et documents distincts, mais si conflit, résolvez-le comme traité précédemment*
7. supprimez la branche annexe qui est maintenant inutile
   * git branch -d [...]

### Développez en collaboratif - partie 3 (branches annexes)

1. **récupérez la dernière version du projet**
2. développez dans une branche annexe
   * créez une nouvelle branche et **placez vous dans cette branche**
   * effectuez l’étape I
   * créez une nouvelle version dans cette branche
3. téléversez la version **de votre branche** annexe sur GitHub
   * git push origin [...]
4. fusionnez la branche annexe à la branche principale sur GitHub
   * effectuez sur GitHub la demande de fusion
     + Pull request
   * réalisez la fusion sur GitHub
     + Merge pull request
5. Mettez à jour le code sur votre ordinateur
   * **placez vous dans la branche principale**
   * **récupérez la dernière version du projet**