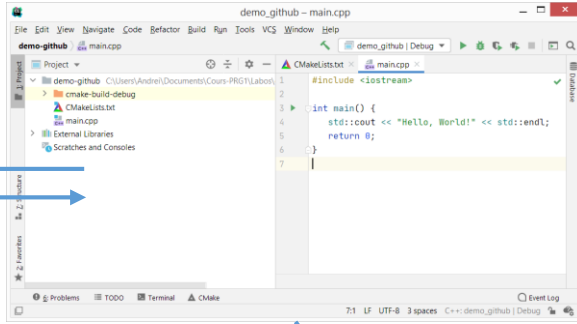


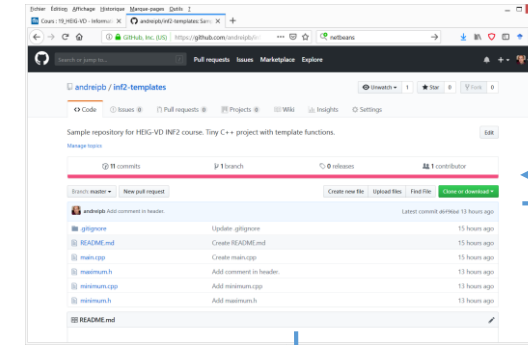
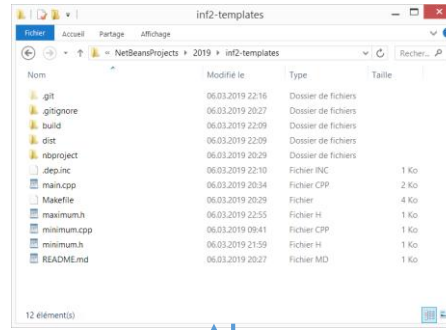
IDE (CLion) + Git

OS local (Windows)

Serveur GitHub, browser



Modifier, compiler et exécuter des projets en C++ (ajouter, modifier fichiers .h et .cpp)



Ajouter, modifier ou supprimer des fichiers .h ou .cpp

+télécharger comme zip

Pour chaque utilisateur ayant accès au dépôt (*repository*) sur GitHub

save / load

(add et) commit

push

clone (1ère fois),
update (=pull)

Client Git local :
gère les versions locales des fichiers (grâce à un dossier caché .git)

Serveur GitHub :
gère les versions des fichiers dans le dépôt partagé

Pour chaque utilisateur, en fonction des droits

«Version Control» avec GitHub pour les labos de PRG1

1. Tous les membres du groupe se créent des comptes sur GitHub.
2. Cadre général : l'un-e des étudiant-e-s crée un *repository* (de préférence privé) à son nom et donne l'accès (read/write) aux autres
 - au début, le projet peut être presque vide : on ajoute un fichier README.md et un fichier .gitignore (choisir celui par défaut pour C++)
 - on peut aussi ajouter un premier fichier .cpp pour tester
 - deux façons d'ajouter de nouveaux fichiers à ce *repository*
 - via l'interface web de GitHub, en collant le code C++ dans le fichier, puis **pull**
 - via un projet local, en poussant (**push**) les fichiers vers GitHub
3. Installer le logiciel Git sur chaque ordinateur individuel (<https://git-scm.com/>) puis activer les *plugins* Git et Github de CLion (avec File > Settings) : les commandes Git sont disponibles sous le menu **VCS**
 - dans File>Settings >VersionControl>Git indiquer le *git.exe* correct
 - dans File>Settings >VersionControl>GitHub indiquer *votre compte*
4. Pour connecter le projet GitHub avec un projet local sur CLion (de chaque contributeur), chacun crée une version locale avec « *Get from Version Control* » (menu VCS ou écran d'accueil)
 - **important** : ici, on ne va pas gérer plusieurs version parallèles d'un projet (*branches*) -- on considère la seule branche *master* qui aura des versions successives (pourtant les branches sont un concept important)
 - **CLion** récupère le projet depuis GitHub (quand on clique sur **clone**) et crée une version locale, dans un dossier dont vous choisissez le nom
 - il faut donner correctement l'URL du projet de GitHub, ainsi que le nom d'utilisateur et le mot de passe
 - tous les projets n'ayant pas de fichier CMakeLists.txt (nécessaire à CMake sous CLion) il peut être nécessaire d'en créer un manuellement et l'ajouter au projet
 - à ce stade, si les sources sont complètes sous CLion, le projet doit





pouvoir compiler, puis être exécuté localement

- cela finit la phase de mise en place du projet sur GitHub et en local

5. Procédure de travail courante

- s'assurer (avec ses collègues) que l'on travaille toujours sur la version la plus récente d'un fichier, sinon, il y aura des conflits entre modifications
- utiliser souvent la commande **update** (≈ pull) qui ramène les nouveautés depuis GitHub (flèche bleue ou menu VCS ou Ctrl+T) et les intègre (**merge**) à vos fichiers
- modifier des fichiers localement (avec CLion) + tester tout le projet
- marquer une version locale avec **commit** (coche verte ou Ctrl+K)
- envoyer versions locales vers GitHub avec **push** (flèche verte, Ctrl+Maj+K)
 - attention : si le repo distant a changé, impossible de faire **push** ! On doit alors faire **pull**, résoudre les conflits localement, puis refaire **commit** et **push**
- toujours faire **update** pour récupérer les n^les versions depuis GitHub
- création en local d'un nouveau fichier : il faut d'abord l'ajouter avec **VCS > add** aux fichiers du projet connus de Git, puis **commit** et **push**
- vous pouvez aussi créer un fichier avec l'éditeur web de GitHub (écrire le code C++), puis chaque étudiant-e récupère la copie avec **update**

6. Comparaison de versions

- sur GitHub.com, cliquer sur l'historique  puis sur le bouton  pour voir l'état du repo à un ancien commit
- sur CLion : boutons   «show history» | «rollback», ou fenêtre Git (Alt+9)

7. Conclusion : vous pouvez vous limiter à quatre commandes (une fois que le projet est mis en place sur GitHub et localement avec **clone**), qui sont : **add** → **commit** → **push** → **update**



Exemple : <https://github.com/andreipb/inf2-templates>
Livre : <https://git-scm.com/book/en/v2> (chapitres 1 et 2)
Offre éduc. : <https://education.github.com/pack>
☺ <https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf>