Aide-mémoire GIT (gitlab.com)

GitLab est un service qui permet de créer des dépôts en ligne, ce qui permet de les partager avec d'autres personnes. Il faut créer un compte sur https://gitlab.com/ si ce n'est pas déjà fait.

Créer un projet sur gitlab.com

1. Télécharger git...
2. Créer un compte GitLab (RAS).
3. Valider l'adresse mail en cliquant dans le mail reçu à l'adresse donnée.
4. Créer une équipe.
5. Créer un repository (privé pour qu'il ne soit pas visible sans être invité).
6. Ajouter une clef ssh :
   - sous Linux:
     ```
     ssh-keygen -t rsa -b 2048 -C "mon-email@mon-univ.fr"
     ```
   - sous Windows, c'est la même commande, mais il faut lancer GIT Bash :
     ```
     ssh-keygen -t rsa -b 2048 -C "mon-email@mon-univ.fr"
     ```
7. Si le projet avait déjà commencé en local :
   ```
   (mkdir /path/to/your/project)
   cd /path/to/your/project
   (git init)
   git remote add origin git@gitlab.com:myname/myproject.git
   ```
   On peut alors créer/modifier un fichier, faire un commit et un push :
   ```
   echo "My name" >> contributors.txt
   git add contributors.txt
   git \commit\ -m 'Initial \commit\ with contributors'
   git push -u origin master
   ```
8. Si le projet n'avait pas déjà commencé en local :
   ```
   git clone git@gitlab.com:myname/myproject.git
   cd projet-bimb-0
   touch README.md
   git add README.md
   git commit -m "add README"
   git push -u origin master
   ```
9. Envoyer des invitations pour partager le projet *via* l'interface GitLab.

Création des comptes de chacun des participants

1. chacun crée un compte sur GitLab.
2. chacun crée un dossier GIT
   ```
   mkdir /path/to/your/project
   cd /path/to/your/project
   git init
   ```
3. chacun clone le projet initié
   ```
   git clone git@gitlab.com:myname/myproject.git
   git clone https://gitlab.com/myname/myproject.git
   ```
   L'adresse est à récupérer sur son propre compte GitLab (après avoir reçu l'invitation).

Ajouter les modifications à l'index

```
git add <fichier1> <fichier2> <fichier3> ...
```

On peut indexer tous les fichiers modifiés qui sont déjà suivis avec :

```
git add -u
```

mais les nouveaux fichiers non suivis ne seront pas pris en compte, il faut utiliser la première syntaxe pour les ajouter. Il est aussi possible de manipuler les fichiers avec `git rm` et `git mv` qui font la même chose que `rm` et `mv`, mais prennent en compte ces changements dans l'index.

Ne pas hésiter à réutiliser `git status` pour voir si on a bien ajouté tout ce qu'on voulait. Les fichiers ajoutés à l'index apparaissent en vert.

Valider les modifications

```
git commit -m "<message de commit>"
```

cette commande crée un point de sauvegarde : toutes les modifications sont validées **localement**. Le message de commit est une description rapide des modifications apportées. Il est possible de voir l'historique des commits avec la commande :

```
git log
```

Pour publier

```
git push origin master
git push -u origin master  // pour que ensuite l'appel soit:
git push
```

Pour récupérer des commit partagés

```
git pull origin master
git pull -u origin master  // pour que ensuite l'appel soit:
git pull
```