Université de Bordeaux
Semestre 2

2021-2022

TD6 : GIT
TOUS POUR UN ET UN POUR TOUS

Compétences

- Utiliser Git
- Mettre son projet sur GitLab
- Collaborer à plusieurs

Les fondamentaux

Git est un logiciel libre de gestion de versions décentralisé et créé par Linus Torvalds, le créateur du noyau Linux. Une source d'information et de documentation sur Git est disponible en ligne :

- http://git-scm.com/book/fr/v2
- http://git-scm.com/docs

REMARQUE

Git va probablement vous paraître difficile (voire obscur), je vais essayer de faire en sorte que non. Cependant, c'est aujourd'hui un standard dans le monde du logiciel libre et bien au-delà dans le secteur du numérique ! Nous allons l'utiliser tout au long de cette année, donc adoptez-le dès maintenant et acceptez de perdre du temps pour apprendre à maîtriser cet outil important.

Le langage de Git

Les interactions avec Git se font uniquement en ligne de commande à travers le terminal. Les commandes seront toutes plus ou moins de la forme :

```
git commande paramètre1 paramètre2 --option
```

Si certaines fonctions de `git` ne possèdent pas de paramètre comme `git status` d'autres nécessitent un ou plusieurs paramètres comme `git commit -a --message = "first commit"`.

Configurer Git

Commençons par indiquer à Git qui vous êtes. Cela vous permettra d'être identifié (nom, e-mail) lors de votre collaboration à un projet Git. Dans un terminal, tapez les commandes suivantes en prenant soin d'indiquer correctement votre prénom, votre nom et votre e-mail **académique**.

```
1 $ git config --global user.name "Prénom Nom"
2 $ git config --global user.email "prenom.nom@etu.u-bordeaux.fr"
3 $ git config --global pull.rebase false
```

Vérifiez que tout est correct avec la commande :

```
1 $ git config --global -l
```

Attention : Cette configuration est particulièrement importante pour les évaluations automatiques que nous allons effectuer par la suite dans Moodle ! Chaque étudiant devra soumettre son travail personnel avec son nom d'auteur bien configuré, sinon sa contribution au travail d'équipe ne sera pas comptabilisée !

Créer un nouveau projet

Voilà nous y sommes, nous allons créer notre premier projet Git. Pour l'exemple nous l'appellerons **`firstProject`**. Dans un terminal :

- créez un répertoire **`firstProject`** avec la commande `mkdir`.
- Déplacez-vous dans ce répertoire avec la commande `cd`.
- Pour initialiser votre nouveau **`repository`** Git, tapez la commande `git init`.

Un tout nouveau projet Git vide sur votre ordinateur vient d'être créé. Il est vide à deux égards : il ne contient aucun fichier et aucun commit.

INFORMATION

A Git repository is the `.git/` folder inside a project. This repository tracks all changes made to files in your project, building a history over time. Meaning, if you delete the `.git/` folder, then you delete your project's history.

Commiter

Sur le plan sémantique, chaque commit représente une *image* complète de l'état de votre projet à un instant donné, ce que l'on pourrait traduire par *snapshot*. Git associe à cet état un identifiant unique (appelé *hash de commit*) qui permettra de distinguer cet état de tous les autres.
Une des particularités de Git est qu'il procède uniquement par **addition**. Quand vous supprimez ou modifiez un fichier, vous ajoutez un commit. Les éléments de la base de données Git sont **inaltérables**, c'est-à-dire qu'ils ne peuvent jamais être modifiés, seulement augmentés. **Git est un système d'accumulation** afin que le retour en arrière soit toujours possible.

Le statut de votre projet

- Avec un terminal, placez-vous dans le répertoire `firstProject`, tapez la commande `cat > main.py` puis appuyez sur les touches **Ctrl + D**. Que se passe-t-il ?
- Nous allons maintenant ajouter une instruction à ce fichier directement depuis le terminal. Tapez la commande : `echo "print('Hello World')" > main.py`
- Tapez ensuite `git status`. Nous pouvons observer qu'un fichier a été ajouté avec le statut **Fichiers non suivis**

Avant de pouvoir commiter un fichier, celui-ci doit avoir le statut **Nouveau fichier**, il faut donc l'ajouter à la base de données de Git avec la commande `git add main.py`. Vérifiez ce qu'il s'est passé avec la commande `git status`, commentez.

Vous venez d'**indexer** votre fichier `main.py`, il ne vous reste plus qu'à le commiter : `git commit -am "first commit"`.

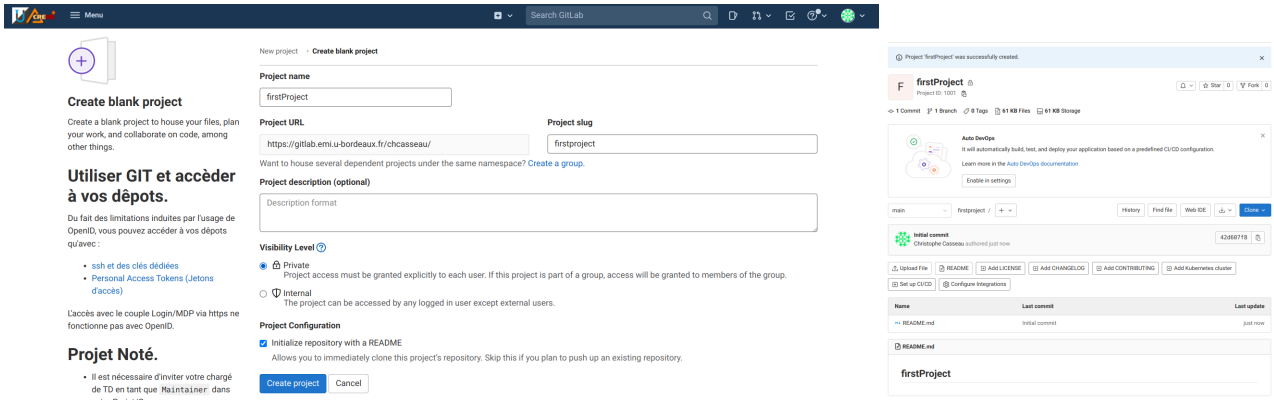
1. Ajoutez (toujours en ligne de commande) une deuxième ligne à votre fichier.
2. Observez le statut de votre fichier.
3. Effectuez un commit avec un commentaire différent.
4. Tapez la commande `git log` et commentez le résultat.

Repository distant

Jusqu'à maintenant toutes les modifications que vous avez effectuées résidaient dans un seul et même endroit, votre **ORDINATEUR**. Tout votre travail se trouve donc dans un **dépôt (*repository*) local** sur le support de stockage de votre ordinateur. Mais la plupart du temps, Git est utilisé pour collaborer avec d'autres. Nous allons donc effectuer une copie de votre projet Git sur un **repository distant** qui se trouve en dehors de votre ordinateur. Pour faire cela, il existe plusieurs solutions mais les deux principales sont les services en ligne proposés par GitHub et GitLab. Dans notre cas, nous utiliserons le GitLab du CREMI. C'est une centralisation du projet, vous et votre équipe conservez une copie commune du projet sur un serveur distant (que j'appellerai le *lab*) accessible à tout moment par n'importe lequel des membres de l'équipe. Il sera même possible pour un nouveau membre de l'équipe de **cloner** le repository existant à l'aide la commande `git clone <adresse du projet>`.

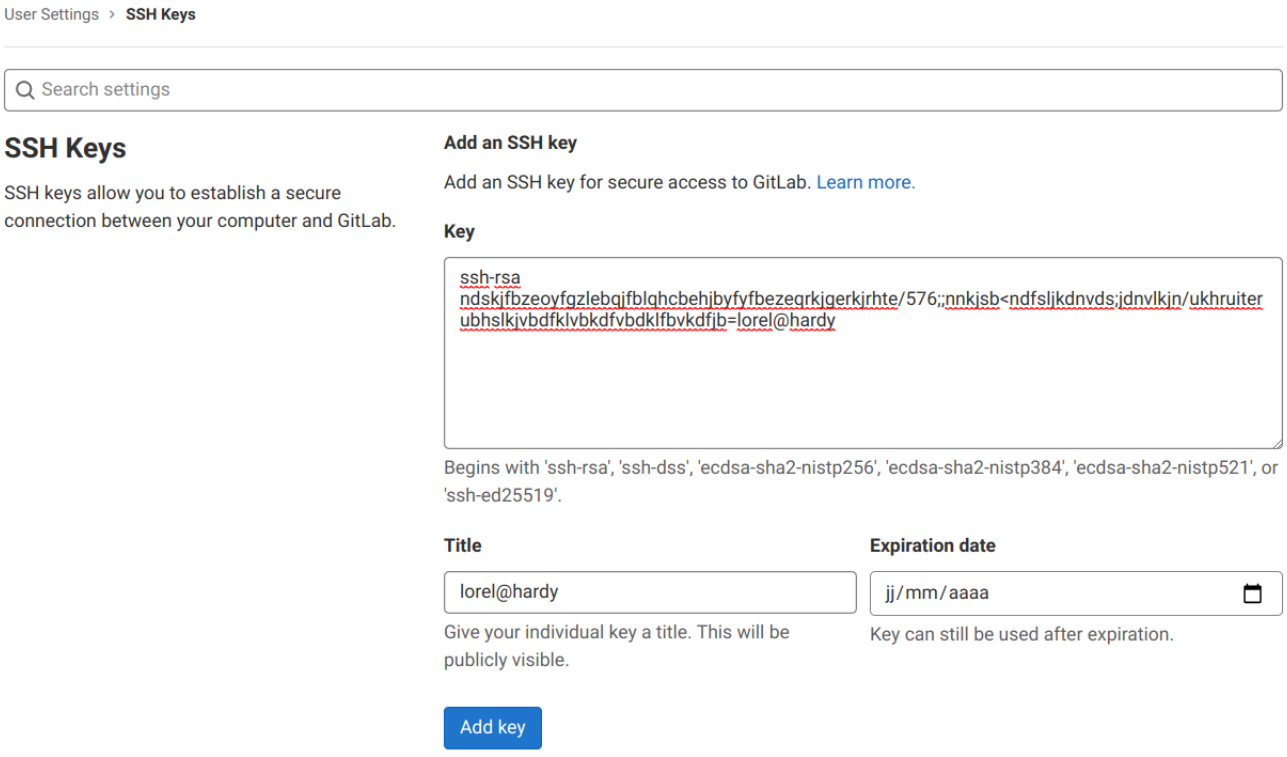
Création d'un projet Git avec GitLab

A l'aide d'un navigateur web, connectez-vous à la plateforme GitLab du CREMI, qui va vous permettre de créer des projets Git et de les administrer tout au long de vos études en Informatique à l'Université de Bordeaux.
Dans cet exercice, vous devez créer un nouveau projet Git du nom de votre choix, par exemple *firstProject*, en choisissant *private* comme visibilité du projet. Attention, un projet *public* est visible de tous ! Choisissez d'initialiser votre projet avec un fichier `README.md`, c'est toujours une bonne idée.



Ajouter une clé SSH

Afin de récupérer une copie locale de son projet Git sur votre machine (ou vos machines), il va être nécessaire d'ajouter votre clé publique dans les *Settings* de votre compte GitLab, également accessible directement à l'URL https://gitlab.emi.u-bordeaux.fr/-/profile/keys.



Normalement vous devriez recevoir un e-mail du CREMI signalant la bonne réception de votre clé SSH.
Si vous avez déjà généré votre clé publique, vous devez disposer d'un fichier `~/.ssh/id_rsa.pub`, dont il faut recopier le texte dans votre navigateur web. Vous pouvez recommencer si vous disposez de plusieurs clés SSH. Si vous ne disposez pas encore d'une clé publique, vous pouvez en créer une nouvelle sur votre machine au CREMI et/ou à la maison avec la commande `ssh-keygen` (https://docs.gitlab.com/ee/ssh/).

Obtenir un repository local

L'étape suivante va consister à récupérer une copie locale du projet Git en utilisant la commande `git clone <repository>`. Pour trouvez l'adresse de votre dépôt, cliquez sur le bouton "Clone with SSH" (bouton bleu, en haut à droite). Attention, il faut absolument effectuer le clone avec la méthode SSH, car la méthode HTTPS n'est pas disponible au CREMI!
En principe, si votre clé SSH est correctement installée, il n'est pas nécessaire de taper votre mot de passe du CREMI, mais uniquement la *passphrase* qui protège votre clé privée SSH (si vous en avez une). Si vous en avez marre de toujours

taper votre mot de passe, pensez à configurer un agent SSH. Un répertoire test est alors créé, qui contient le fichier `README.md` (ajouté à l'initialisation) et un sous-répertoire caché `.git/`

1. Ajoutez un fichier dans votre repository local.
2. Faites un commit.
3. Pour *pousser* votre repository local vers votre repository distant, utilisez la commande `git push`.
4. Pensez à utiliser systématiquement les commandes `git log` et `git status` pour afficher l'état courant de votre dépôt Git.
5. Vérifiez sur la page GitLab de votre projet que la modification a bien été prise en compte. Retrouvez l'identifiant de votre commit, le commentaire associé et vérifiez que le nom d'auteur est correct.
6. Modifiez maintenant le fichier `README.md` directement depuis votre repository distant en y ajoutant du texte de votre choix.
7. Synchronisez votre repository local à l'aide de la commande `git pull`.
8. Vérifiez que tout s'est fait correctement.

Rendu sur Moodle

Ajoutez dans votre projet Gitlab votre chargé de TD, ainsi que l'utilisateur **Moodle Manager** (`_mmoodle`). Pour cela, il faut choisir sur la page web de votre projet le menu `Project Information > Members`, puis inviter les utilisateurs demandés en donnant le rôle **Maintainer**.

Attention : Le choix de ce rôle est important, car il est nécessaire au bon fonctionnement des évaluations automatiques sur Moodle !

Le repository distant du projet

Cet exercice est à réaliser avec les membres de son équipe.

- Pour continuer, il faut qu'un étudiant leader de l'équipe (et un seul) pour créer le projet sur gitlab.
- L'étudiant leader doit ensuite ajouter tous les étudiants membres de son équipe. Pour ce faire, sélectionner le menu `Project Information > Members` dans votre projet, puis inviter les autres étudiants en tant que Maintainer. (Attention : il faut que chaque étudiant se soit connecté au moins un fois à Gitlab pour disposer d'un compte actif.)
- Un des membres doit ensuite `push` la version actuelle de votre projet `VideoTracker`.
- Les autres membres n'ont plus qu'à `clone` le projet

Par la suite il est très important que pour chaque membre le contenu repository local du projet sur l'ordinateur soit le plus proche possible du contenu du repository distant.

Les bonnes pratiques

Avant de modifier le projet sur le repository local vous devez commencer par :

- `git pull`
- `git log`

Une fois votre travail terminé, pour mettre à jour le repository distant

- `git status`
- `git commit`
- `git push`

1. À l'aide du lien suivant https://git-scm.com/docs/git-pull/fr, expliquer le fonctionnement de `git pull`
2. Quel est l'intérêt de faire un `git log` ?
3. Quels renseignements importants vous donnent `git status` ?
4. Que peut-il arriver au moment du `push` si vous avez oubliez de `pull` avant de modifier votre repository ?