Systèmes d'exploitation (SYE)

Profs Daniel Rossier, Alexandre Corbaz, Fiorenzo Gamba
Assistants : Anthony Jaccard, Basile Cuneo, Jean-Pierre Miceli

Git workflow et séparation kernel space – user space

lab01 (Semaine du 18.09.2023)

Objectifs de laboratoire

Il s'agit d'un laboratoire d'introduction à l'environnement de travail pour les laboratoires SYE. Dans ce cadre, chaque étudiant-e aura l'occasion de se familiariser avec les commandes git nécessaires pour créer le dépôt git qui sera utilisé pour les laboratoires tout au long du semestre à partir d'un dépot tiers, récupérer une nouvelle branche à partir du dépôt d'origine, commiter les modifications effectuées dans le cadre du laboratoire et publier la branche pour correction.

Validation du laboratoire

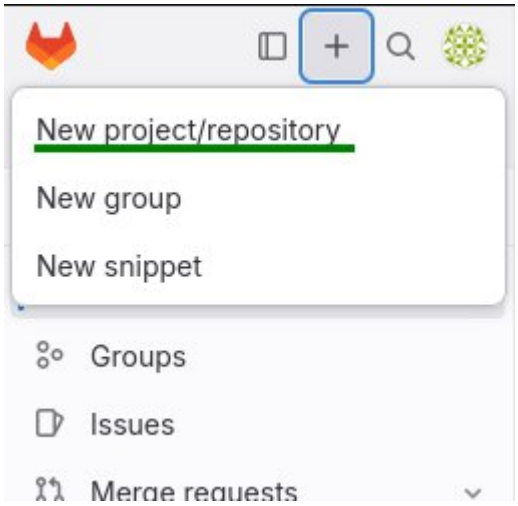
Ce laboratoire détaille les étapes requises pour la validation du laboratoire. Ces étapes seront reprises dans les laboratoires suivants pour les différents rendus.

Etape 1 - Récupération des sources, environnement VSCode et séparation kernel/usr

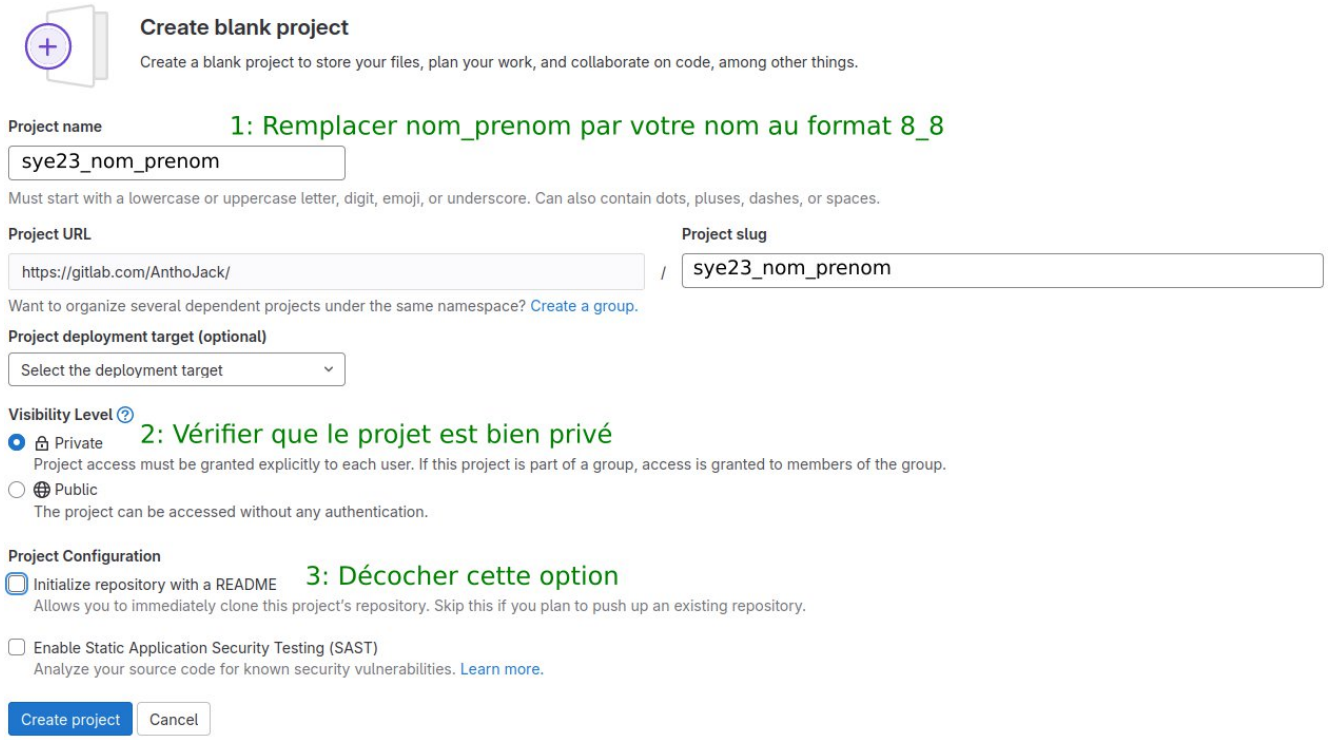
La première étape à effectuer permet de récupérer l'environnement complet nécessaire aux laboratoires SYE.

Afin d'éviter de devoir saisir son nom d'utilisateur et mot de passe à chaque communication avec gitlab, nous vous recommandons d'enregistrer une paire de clés SSH sur votre compte gitlab si ce n'est pas déjà fait. Pour générer une paire de clés SSH et les enregistrer sur gitlab, vous pouvez suivre ces instructions .

a) Tout d'abord, il est nécessaire de créer un dépot vide sur votre compte Gitlab. Allez sur votre compte puis cliquez sur la petite croix en haut à gauche et sélectionnez "New project/repository" dans le menu déroulant qui s'affiche

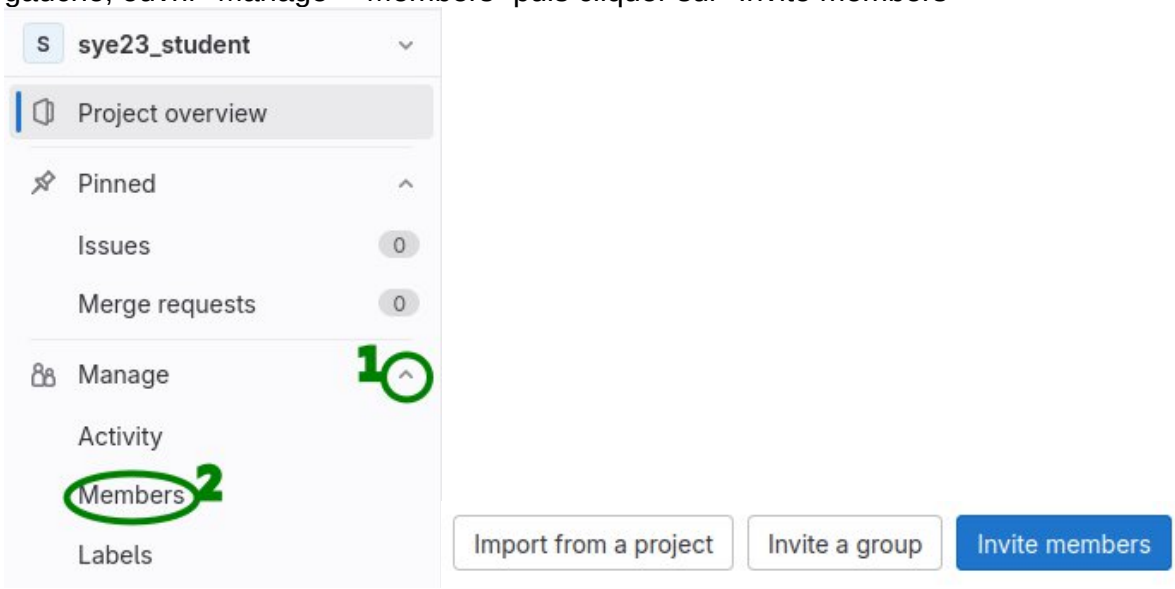


Dans la page qui s'ouvre, choisissez "Create blank project" puis configurez le projet comme indiqué sur la capture d'écran suivante



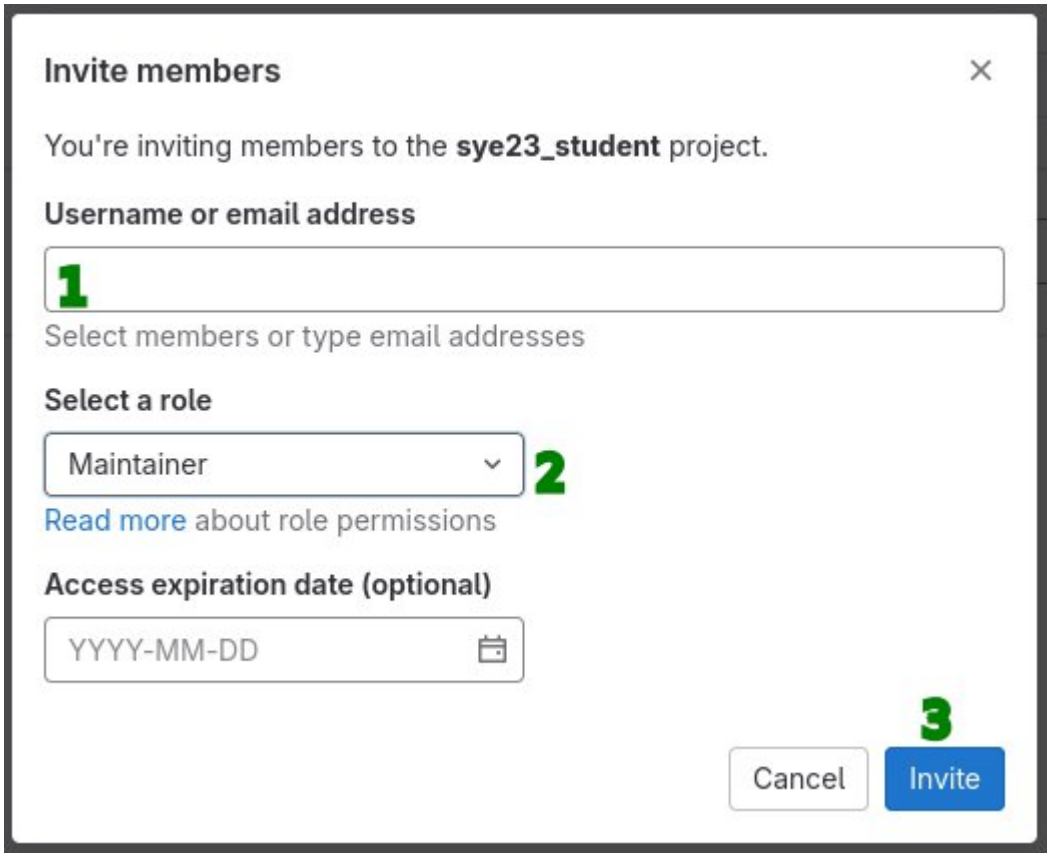
Une fois le projet configuré correctement, cliquer sur "Create project"

b) Afin que nous puissions accéder à votre dépôt pour corriger vos laboratoires, dans la barre à gauche, ouvrir "Manage">"Members" puis cliquer sur "Invite members"

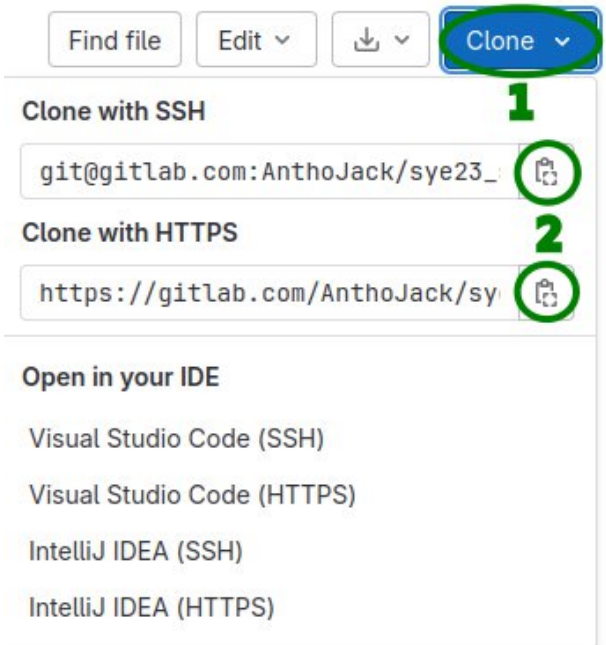


Dans la fenêtre qui s'ouvre entrer le nom d'utilisateur de votre professeur (demandez-le-lui) et sélectionnez le rôle "Maintainer" dans le menu déroulant et cliquer sur "Invite". Faites de même pour tous les assistants avec les usernames suivants:

- Basile Cuneo : bcu-reds
- Anthony Jaccard : AnthoJack



c) Une fois votre propre repository créé et configuré, vous pouvez cliquer sur le bouton "Clone" afin de récupérer le lien SSH ou HTTPS vers celui-ci.



d) Dans un terminal de votre VM, rendez-vous dans le dossier où vous souhaitez cloner votre dépôt et entrez la commande suivante (pour coller le lien dans le terminal, vous pouvez utiliser ctrl + shift + v)

```
reds@reds2023:~$ git clone <SSH or HTTPS link>
```

e) Afin de récupérer le contenu du dépôt de référence, utiliser la commande suivante

```
reds@reds2023:~/sye23_student$ git remote add upstream https://reds-gitlab.heig-vd.ch/reds-public/sye23_student.git
```

Ceci créera un lien vers un deuxième dépôt distant nommé "upstream". Ce dépôt contiendra les branches à récupérer au début de chaque laboratoire. Pour récupérer la branche de ce laboratoire, utiliser les commandes suivantes

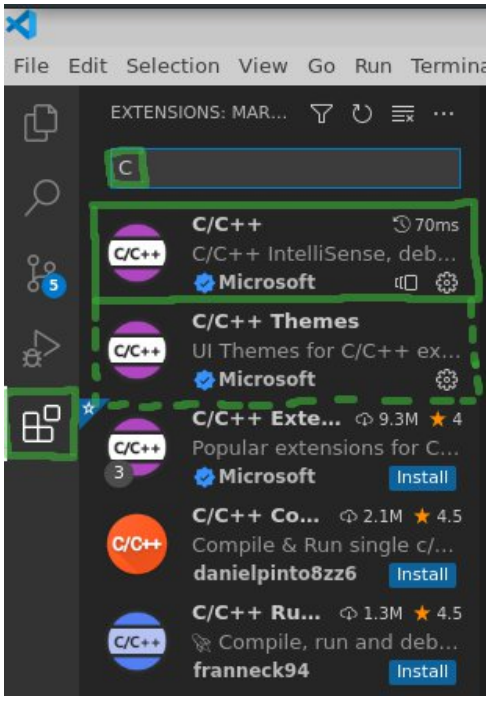
```
reds@reds2023:~/sye23_student$ git fetch upstream
reds@reds2023:~/sye23_student$ git checkout lab01
```

Vous devriez alors disposer de tous les fichiers nécessaires et pouvez donc lancer Visual Studio Code

f) Afin d'ouvrir le répertoire dans VSCode, vous pouvez ouvrir VSCode normalement puis ouvrir depuis l'interface graphique ou alors vous pouvez tapper

```
reds@reds2023:~/sye23_student$ code .
```

g) Pour utiliser le répertoire dans Visual Studio Code et disposer de toutes les fonctionnalités nécessaires pour ce cours, il faut installer l'extension pour le support du C/C++ (Et éventuellement les thèmes pour ce même langage)



La configuration des fonctionnalités est fournie dans le répertoire du laboratoire (dossier caché *.vscode*)

Etape 2 - Séparation kernel/usr

Dans le fichier "rapport.md" répondre aux questions suivantes:

a) Quelle couche de l'OS (kernel/user) est responsable de
   a. Communiquer avec le matériel
   b. Fournir un système de fichier
   c. Afficher une interface graphique
   d. Gérer le temps d'exécution des programmes en cours d'exécution sur le processeur

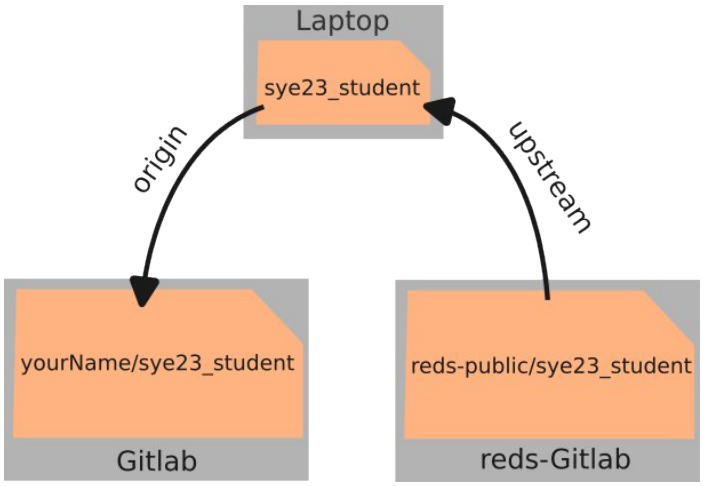
b) Explorez l'arborescence du laboratoire. Quels dossiers de la racine contiennent les fichiers relatifs aux couches kernel/user ?

c) La *libc* est l'implémentation dans Linux de la librairie standard C qui fournit un ensemble de fonctions, structures et autres fonctionnalités pour créer des programmes interagissant avec l'OS (Conseil: référez vous au document "Outils pratiques sur Linux").
   a. Cherchez et donnez le chemin vers le dossier contenant son implémentation dans le répertoire du laboratoire
   b. Dans quelle couche de l'OS se situe-t-elle ? Pourquoi ?
   c. Qu'est-ce que cela implique lors du développement dans l'autre couche de l'OS ?

d) Trouvez le chemin du dossier contenant le code pour le support des architectures processeur (jeu d'instruction) prises en charge par SO3 ? Quelles sont-elles ?

Etape 3 – Commit, tag et push pour rendu

Afin de valider le laboratoire, il est nécessaire de commit les changements effectués et de les envoyer sur votre repo gitlab. Il est possible de faire cela soit dans un terminal, soit dans l'interface graphique de VScode. L'architecture des dépôts pour ce cours est la suivante



Les branches des laboratoires seront le plus souvent récupérées depuis le dépot "upstream" (hébergé sur l'instance gitlab du reds) selon la procédure vue précédemment ("fetch" -> "checkout"). A la fin des laboratoires, vous devrez envoyer la branche avec vos changements sur votre dépôt "origin" sur le site officiel Gitlab avec la procédure décrite ci-après (chaque étape est décrite une fois en version terminal et une fois en version graphique, sur VScode).

a. Afficher les changements effectués

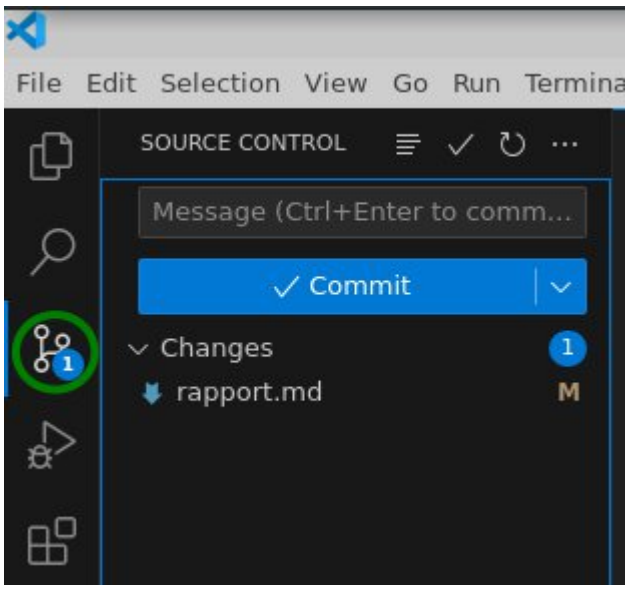
```
reds@reds2023:sye23_student$ git status
On branch lab01
```

```
Your branch is up to date with 'origin/lab01'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   rapport.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Note: la ligne "modified: rapport.md" devrait apparaître en rouge

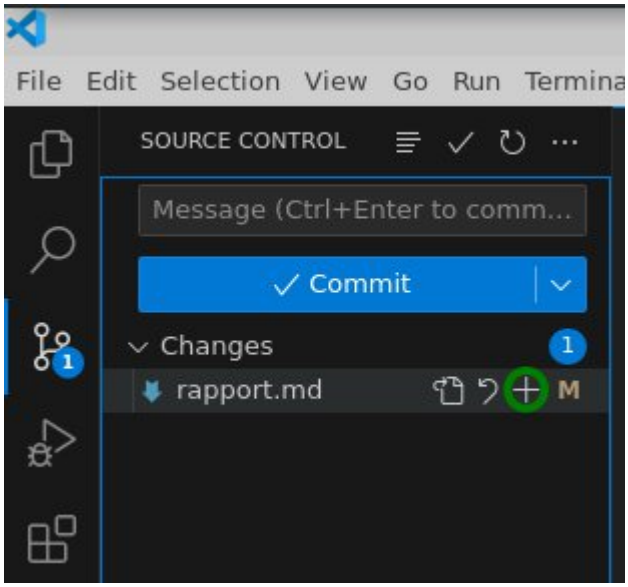


b. Marquer les fichiers à commiter

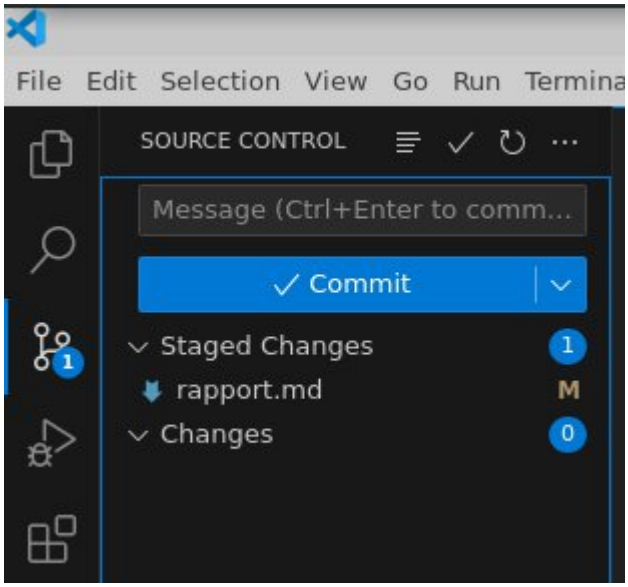
```
reds@reds2023:sye23_student$ git add rapport.md
reds@reds2023:sye23_student$ git status
On branch lab01
Your branch is up to date with 'origin/lab01'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   rapport.md
```

Note: la ligne "modified: rapport.md" devrait apparaître en vert maintenant



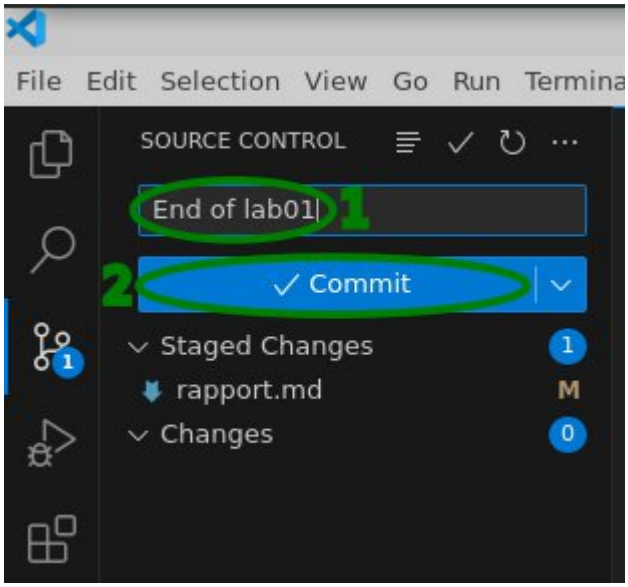
Les icônes supplémentaires apparaissent lors du survol du curseur



c. Effectuer le commit

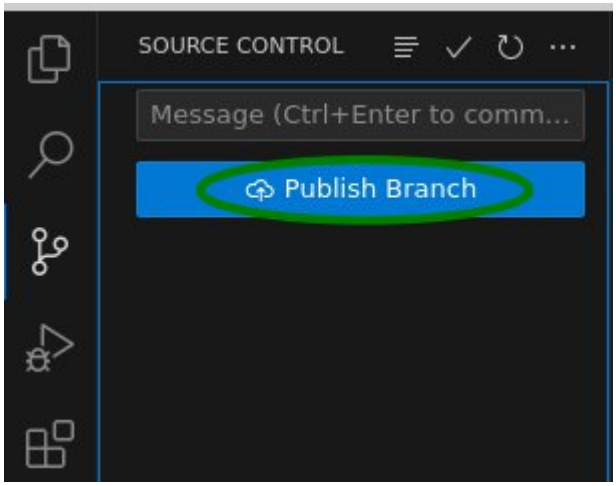
```
reds@reds2023:sye23_student$ git commit -m "End of lab01"
[lab01 19ecf47] End of lab01
 1 file changed, 4 insertions(+)
```

Note: si l’output de la commande git commit ne correspond pas à l’image ci-dessus mais que les informations sur l’utilisateur sont demandées, saisissez les deux commandes demandées en remplaçant “you@example.com” par votre adresse email (sans les guillemets) et “Your Name” par votre nom (également sans les guillements). Une fois ceci fait, saisissez à nouveau la commande de commit.



d. Publier la branche avec le changement sur votre repo Gitlab

```
reds@reds2023:sye23_student$ git push –u origin
```



b) **ATTENTION**: La compilation peut entrainer la génération de fichiers qui ne sont pas ignorés par le *.gitignore* du repo. Afin d'éviter de polluer le résumé du rendu, veillez toujours à ne commiter que les fichiers que vous avez modifié vous-même