

# Introduction au programme



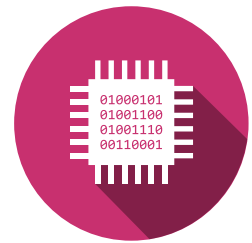
## Table des matières

<b>1 Objectifs</b>	<b>1</b>
<b>2 Introduction</b>	<b>2</b>
2.1 Échange de groupe . . . . .	2
<b>3 Git Repository personnel</b>	<b>3</b>
3.1 Création d'un repository personnel . . . . .	3
3.2 Clonage du repository sur U:\ . . . . .	4
3.2.1 Via ligne de commande . . . . .	4
3.2.2 Via Sublime Merge GUI (optionel) . . . . .	5
3.2.3 Clonage manuel (optionel) . . . . .	6
3.3 Enregistrer le project après chaque laboratoire Github . . . . .	6
3.3.1 Via ligne de commande . . . . .	7
3.3.2 Via Sublime Merge GUI . . . . .	7
<b>4 GIT Commandes</b>	<b>10</b>
4.1 Examen des modification et création d'une opération de validation . . . . .	10
4.2 Synchronisation des changements . . . . .	10
4.3 Commandes Git les plus utilisées . . . . .	10
4.3.1 Start a working area . . . . .	10
4.3.2 Work on the current change . . . . .	10
4.3.3 Examine the history and state . . . . .	11
4.3.4 Grow, mark and tweak your common history . . . . .	11
4.3.5 Collaborate . . . . .	11

## 1 Objectifs

Ce document sert de préparation pour le laboratoire. Il est nécessaire pour recevoir et gérer les dossiers de laboratoire.

La même procédure peut également être utilisée pour les projets.



## 2 Introduction

Les fichiers de laboratoire sont distribués à l'aide de l'outil Github Classroom. Chaque étudiant doit avoir et gérer son propre repo Git.

Git est un outil de gestion des fichiers de code. Il vous permet de les stocker en toute sécurité sur les serveurs Github et de stocker et comparer différents niveaux de révision.

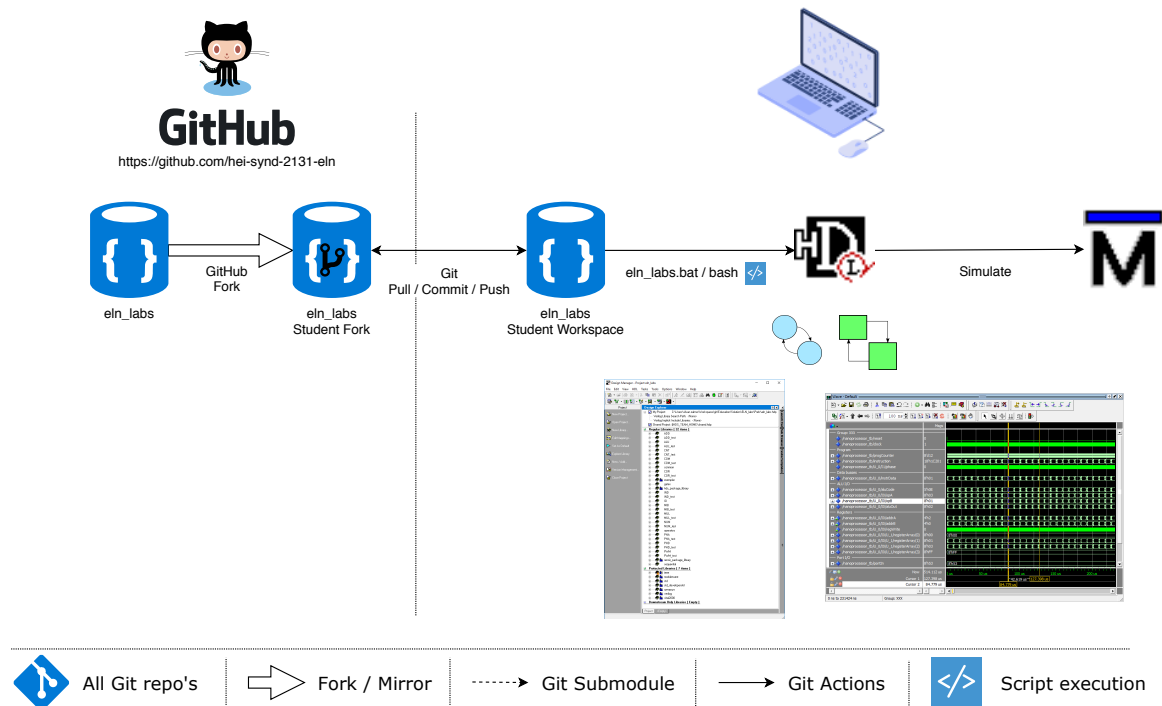
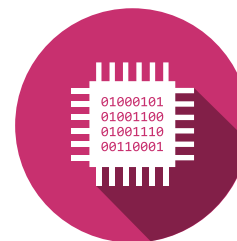


FIGURE 1 – Github système

### 2.1 Échange de groupe

Pendant le laboratoire, vous travaillerez en groupe. Il est important que vous distribuiez les fichiers aux membres du groupe à la fin de chaque laboratoire. Il vous suffit de copier le dossier du laboratoire concerné.



### 3 Git Repository personnel

Un git repository est le nom d'une base de données de code décrite. Les 3 méthodes suivantes expliquent comment transférer le repository fourni sur le disque personnel U:\.

#### 3.1 Création d'un repository personnel

Créez un compte Github avec votre email HEVS

- Visitez le site web : <https://github.com/join>
- Username : <prenom><nomDeFamille>
- Email : <prenom><nomDeFamille>@students.hevs.ch

FIGURE 2 – Join Github page

Suivez le lien d'invitation que vous avez reçu par courriel.

Il a la forme <https://classroom.github.com/<X>/<YYYYYYYY>>

FIGURE 3 – Github Classroom invitation

Accepter l'invitation avec votre compte Github nouvellement créé

Accept this assignment

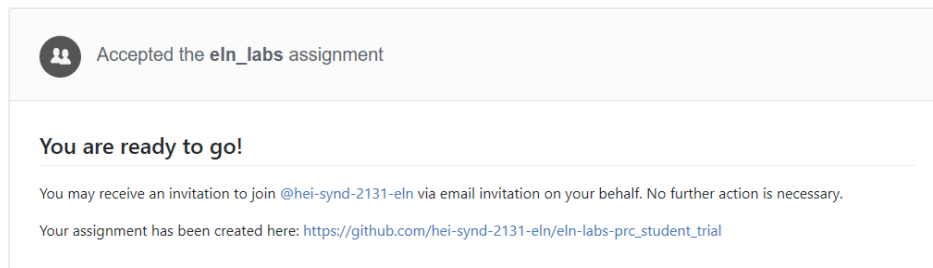
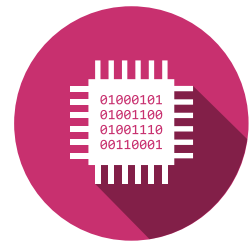


FIGURE 4 – Github Classroom Assignment

Suivez le lien pour accéder à votre propre repository `eln-labs`. Vous y trouverez également le lien pour le clonage.

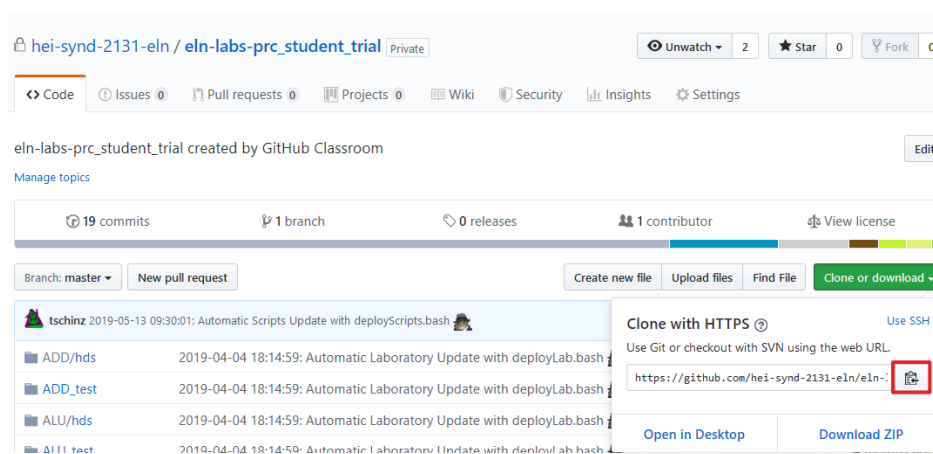


FIGURE 5 – Github Student Repository

## 3.2 Clonage du repository sur `u:\`

### 3.2.1 Via ligne de commande

Ouvrir une ligne de commande

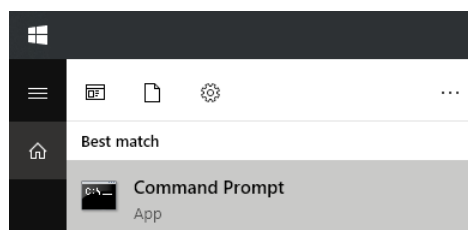
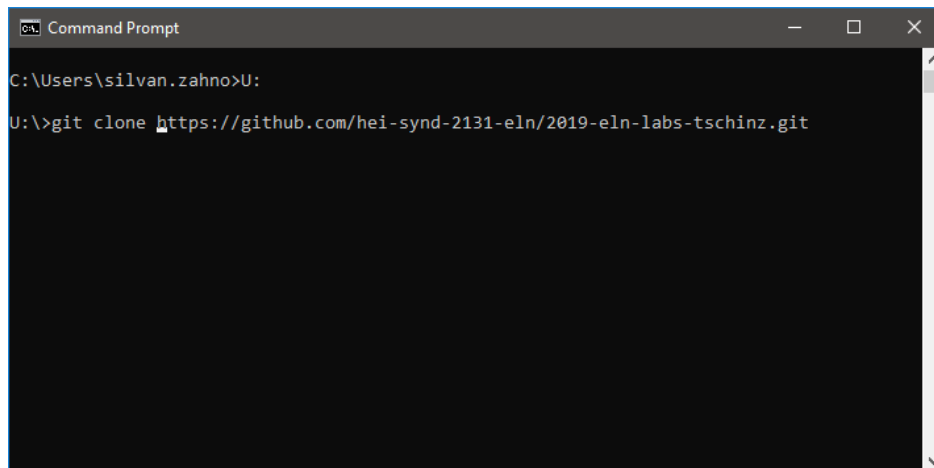
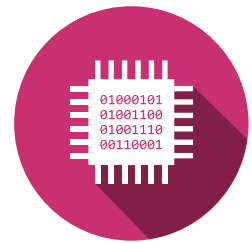


FIGURE 6 – Ouvrir ligne de commande

Exécute la commande `git clone`.



```
Command Prompt
C:\Users\silvan.zahno>U:
U:>git clone https://github.com/hej-synd-2131-eln/2019-eln-labs-tschinz.git
```

FIGURE 7 – Console Windows

U:  
git clone <Repository\_URL>

### 3.2.2 Via Sublime Merge GUI (optional)

Ouvrir le programme Sublime Merge

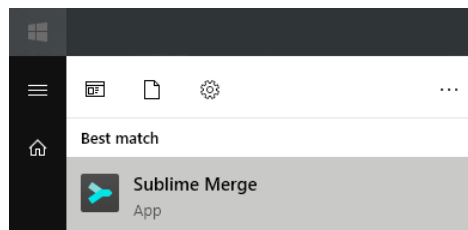


FIGURE 8 – Ouvrir Sublime Merge

File → Clone Repository

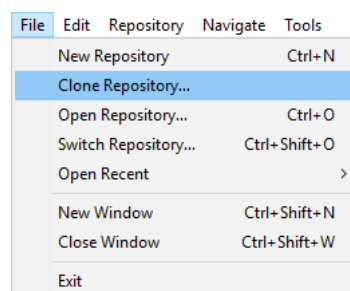


FIGURE 9 – Sublime Merge clonage

Remplit le formulaire Clone Repository

Source URL : de Github

Repository Name : laisser vide

Destination Path : sélectionnez votre répertoire U:\\ personnel

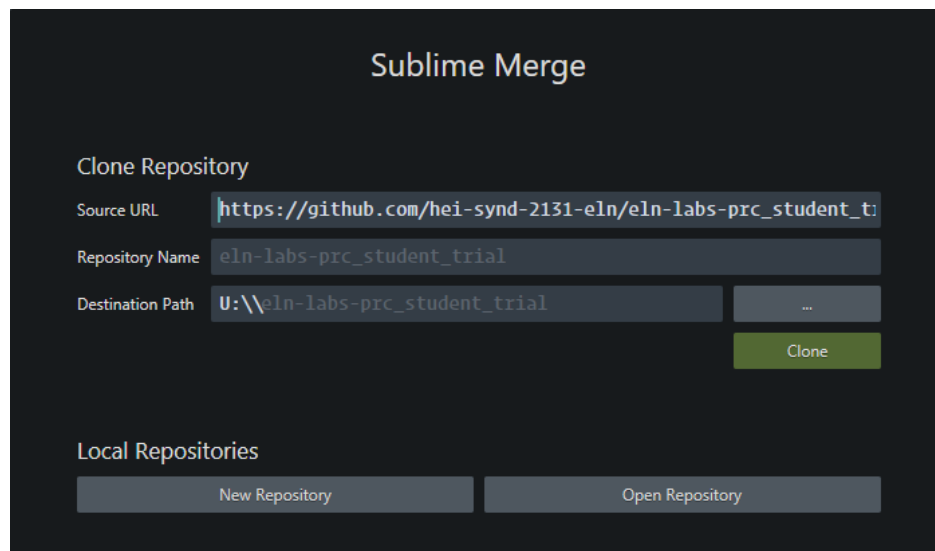
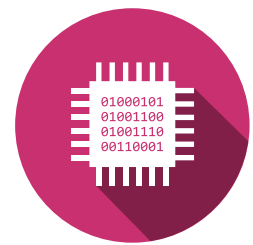


FIGURE 10 – Sublime Merge clonage

Cliquez sur le bouton vert Clone

### 3.2.3 Clonage manuel (optionnel)

Alternativement, le repository peut aussi être téléchargé manuellement sous forme de fichier zip et décompressé dans le répertoire U:\\.

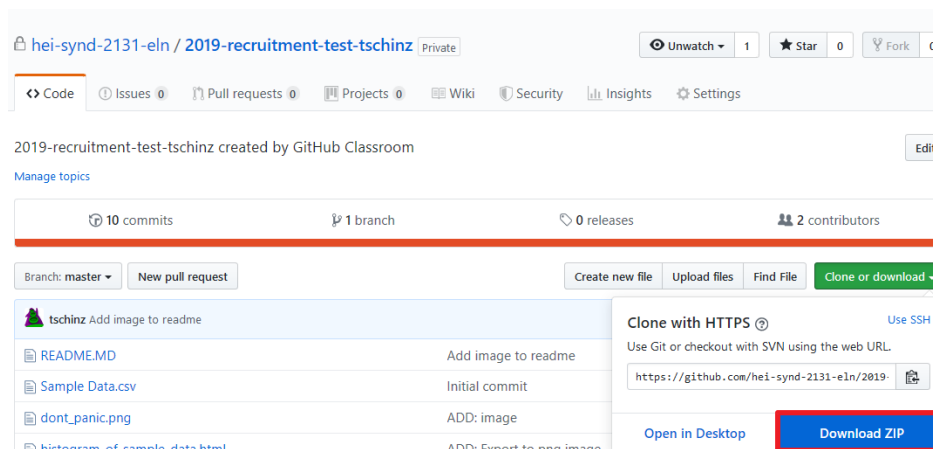


FIGURE 11 – téléchargement manuel

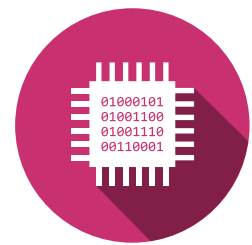
## 3.3 Enregistrer le project après chaque laboratoire Github



Puisque chaque étudiant a son propre repository de git, les membres du groupe doivent échanger le dossier avec le nom du laboratoire respectif entre eux après chaque laboratoire.

De plus, les fichiers modifiés peuvent être téléchargés sur Github. Ceci se fait en 3 étapes :

1. Stage - Sélection des fichiers à sauvegarder
2. Commit – Sauvegarde les fichiers sélectionnés dans le repository local sur le PC



### 3. Push – Synchronise le repository local avec celui sur Github

#### 3.3.1 Via ligne de commande

Exécutez les instructions suivantes dans l'invite de commande.

```
git add -all  
git commit -m "Commit message ex. Labo Num"  
git push
```

#### 3.3.2 Via Sublime Merge GUI

Ouvrir le programme Sublime Merge

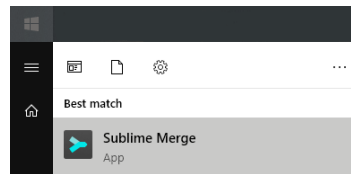


FIGURE 12 – Lancement de Sublime Merge

Sélectionner les fichiers à stage, c'est-à-dire à sauvegarder.

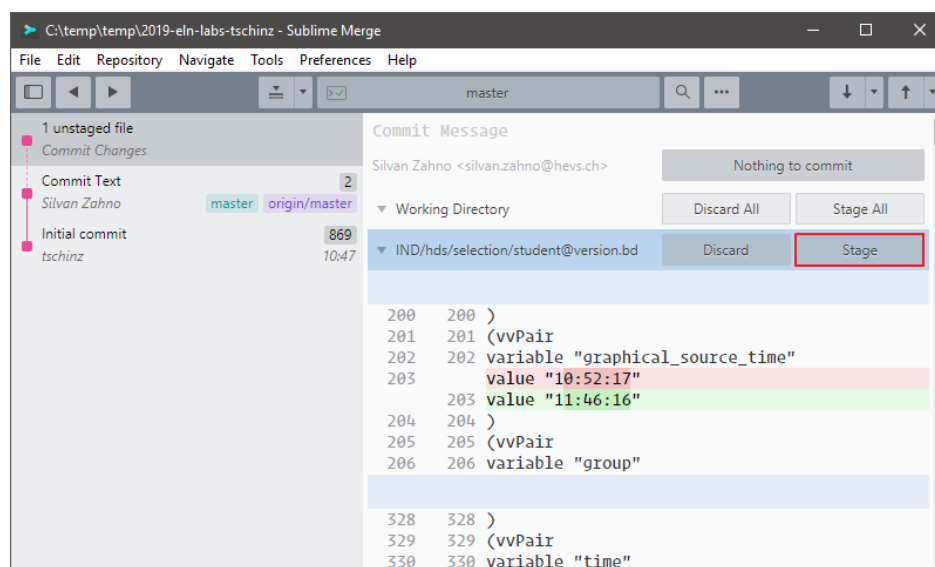


FIGURE 13 – Sélectionner les fichier à stage

Écrire le message de «commit» : Qu'est-ce qui a changé ?

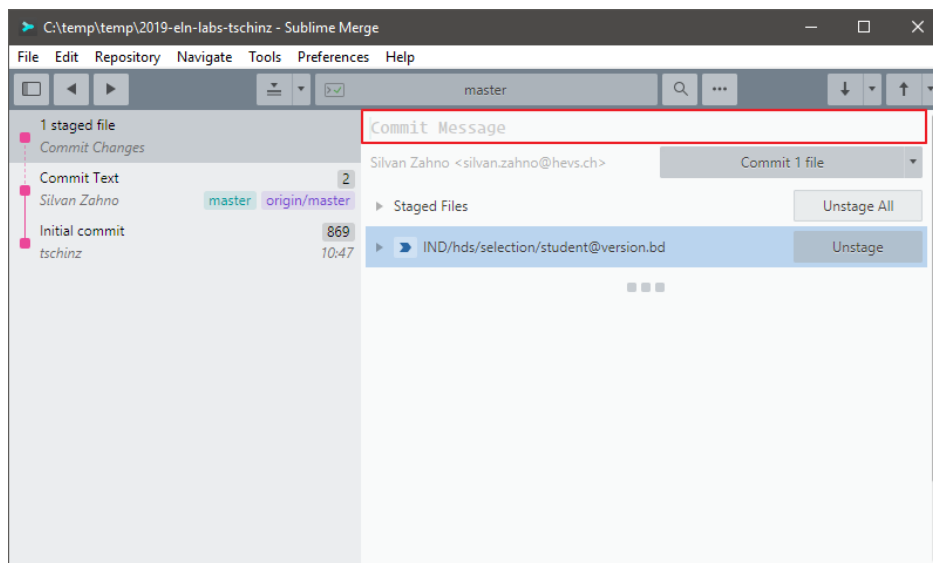
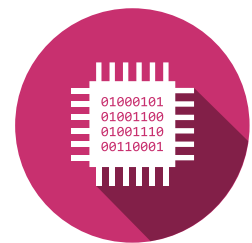


FIGURE 14 – Écrire le Commit message

Valider les modifications, c'est-à-dire les stocker dans le repository local.

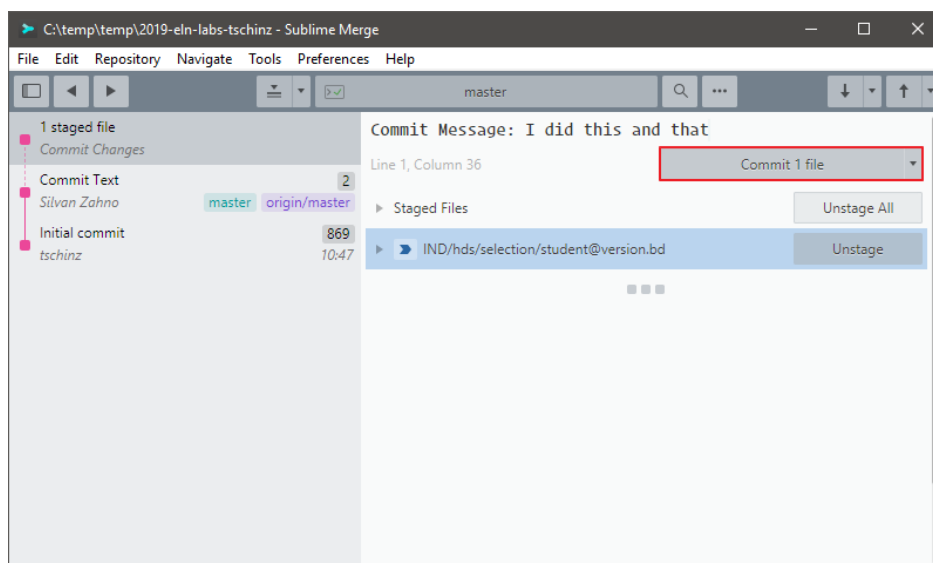


FIGURE 15 – Commit les changement

Push les changements, c'est-à-dire les synchronise avec le repository Github.



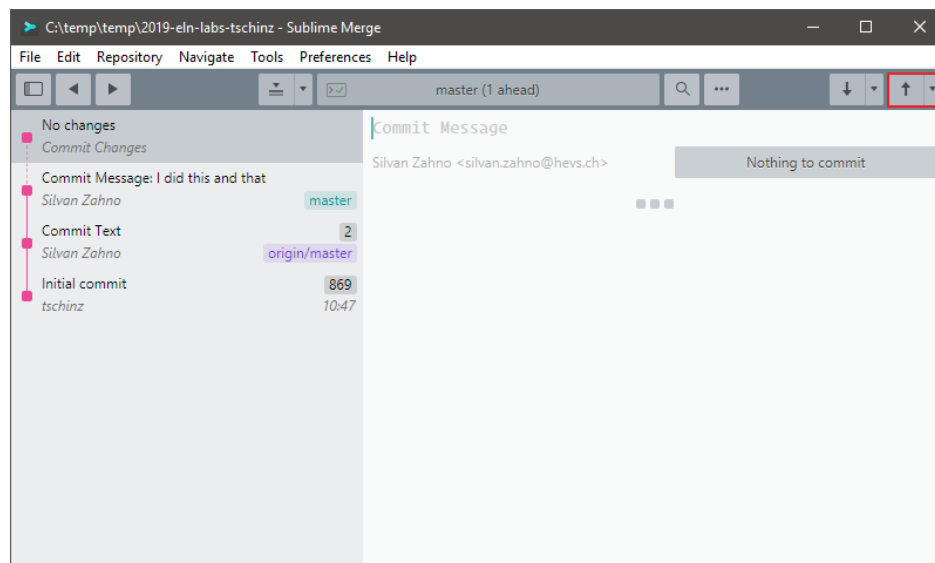
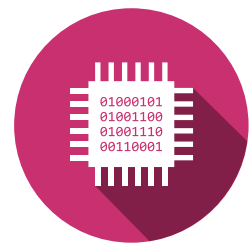


FIGURE 16 – Push les changement

Ensuite vous verrez vos changements localement sur la branche master et sur Github sur la branche origin/master au même niveau.

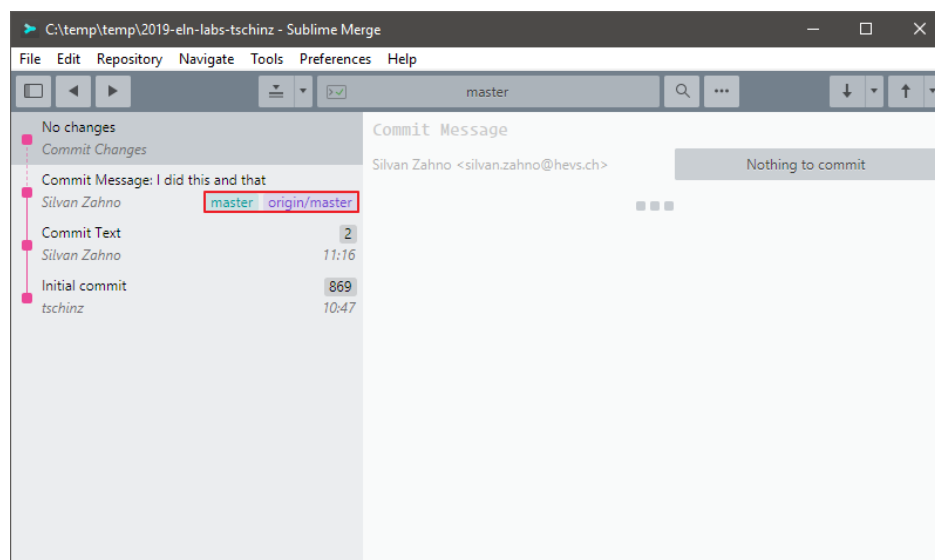
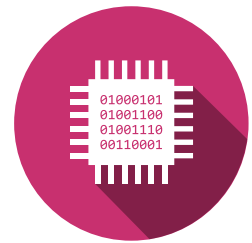


FIGURE 17 – Les changements sont maintenant aussi sur Github



## 4 GIT Commandes

[Github git cheatsheet](#)

### 4.1 Examen des modification et création d'une opération de validation

`git status`

Liste tous les fichiers nouveaux ou modifiés qui sont prêts à être commit.

`git diff`

Affiche les modifications de fichiers qui n'ont pas encore été indexées.

`git add [file]`

Ajoute le fichier au versionning.

`git diff --staged`

Affiche les différences entre l'index ("staging area") et la version actuelle du fichier.

`git reset [file]`

Retire le fichier de l'index ("staging area") mais ne le supprime pas du disque.

`git commit -m "[descriptive message]"`

Inclut tous les fichiers actuellement indexés de façon permanente dans l'historique des versions.

### 4.2 Synchronisation des changements

Enregistrement d'un référentiel externe (URL) et échange de l'historique du repository.

`git fetch [remote]`

Télécharge l'historique complet d'un repository externe.

`git merge [remote]/[branch]`

Intègre la branche externe dans la branche locale.

`git push [remote] [branch]`

Pousse la branchae locale (donc tous les commits de celle-ci) sur GitHub.

`git pull`

Récupération de l'historique du repository externe et intégration des modifications sur le repository local.

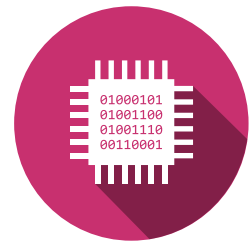
### 4.3 Commandes Git les plus utilisées

#### 4.3.1 Start a working area

- `clone` - Clone a repository into a new directory
- `init` - Create an empty Git repository or reinitialize an existing one

#### 4.3.2 Work on the current change

- `add` - Add file contents to the index
- `mv` - Move or rename a file, a directory, or a symlink



- `reset` - Reset current HEAD to the specified state
- `rm` - Remove files from the working tree and from the index

#### 4.3.3 Examine the history and state

- `log` - Show commit logs
- `show` - Show various types of objects
- `status` - Show the working tree status

#### 4.3.4 Grow, mark and tweak your common history

- `branch` - List, create, or delete branches
- `checkout` - Switch branches or restore working tree files
- `commit` - Record changes to the repository
- `diff` - Show changes between commits, commit and working tree, etc
- `merge` - Join two or more development histories together
- `rebase` - Reapply commits on top of another base tip
- `tag` - Create, list, delete or verify a tag object signed with GPG

#### 4.3.5 Collaborate

- `fetch` - Download objects and refs from another repository
- `pull` - Fetch from and integrate with another repository or a local branch
- `push` - Update remote refs along with associated objects