

# Introduction à la gestion de versions distante avec *git* et *Gitlab*

**Sébastien Jean**

IUT de Valence  
Département Informatique

v2.2, 11 septembre 2025

# Git Vs Gitlab / GitHub / BitBucket

- Git
  - **Système de gestion de versions**
  - **Outil en ligne de commande**
- Gitlab / GitHub / BitBucket
  - **Plateformes d'hébergement** de code versionné
    - En mode **SaaS** (gitlab.com, github.com, bitbucket.org)
- *GitLab* peut aussi être **déployé sur un réseau local**
  - C'est le cas à l'IUT (gitlab.iut-valence.fr)



# Quels services fournissent *Github* & co ?

- **Héberger** un historique (distant)
- **Restaurer** (localement un historique distant
- Intégrer un commit local à l'historique distant
- Intégrer un commit distant à l'historique local
- ...(*le reste plus tard ;-)* )

# Connexion au serveur Gitlab de l'IUT



GitLab Via Ldap      Standard

Username

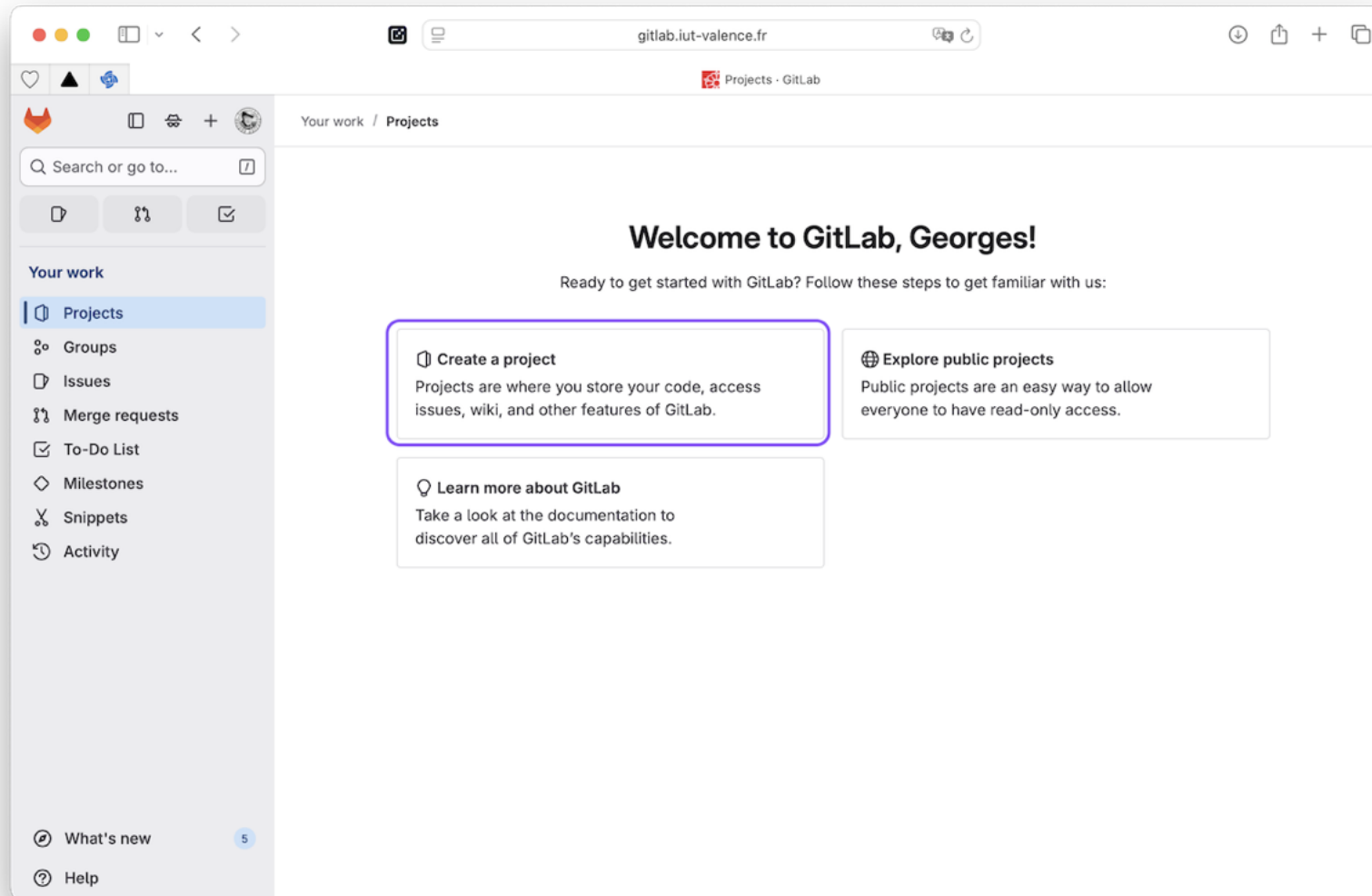
Password

☐ Remember me

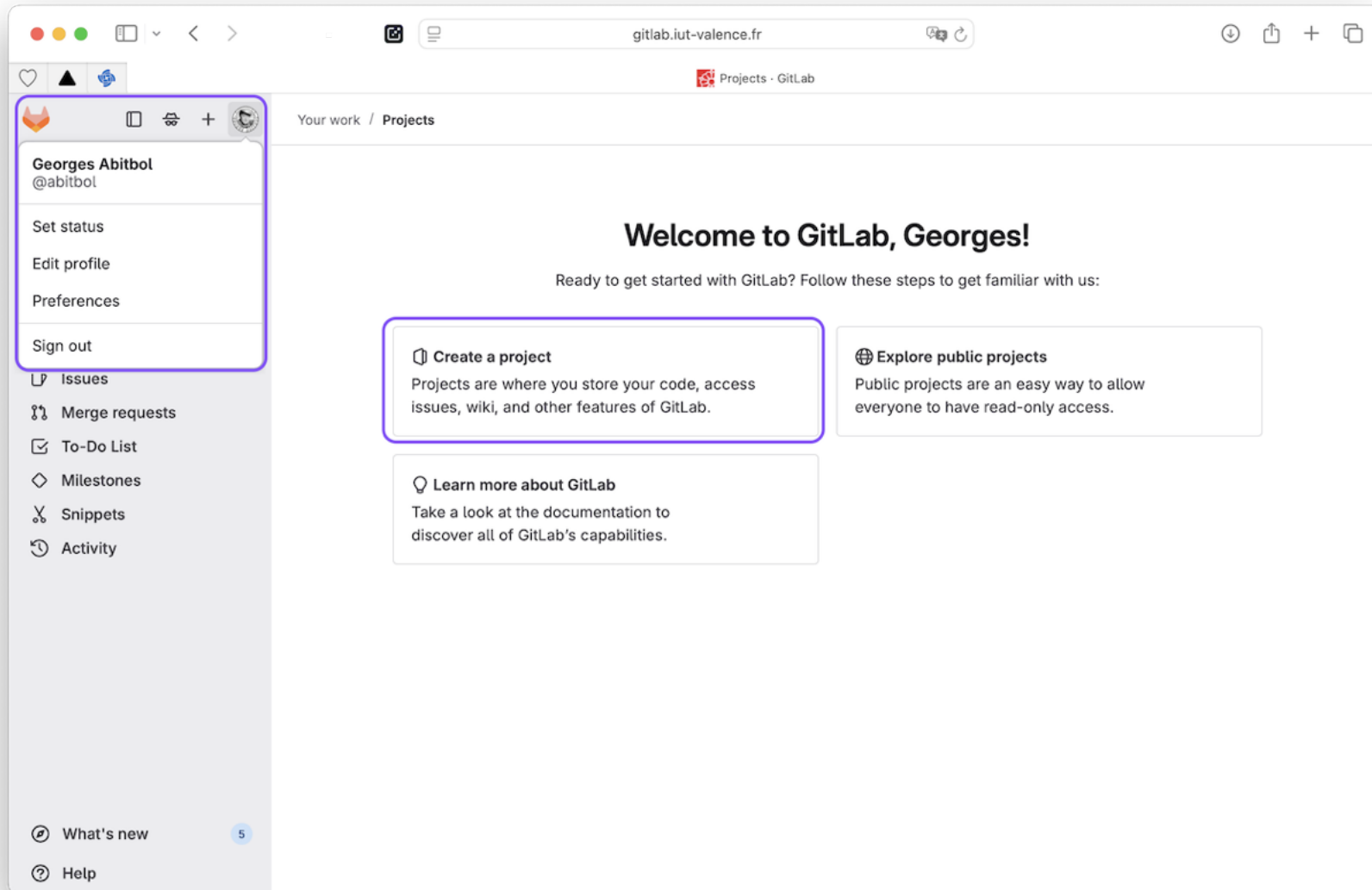
Sign in

- Authentification LDAP avec les identifiants de session Linux/Windows

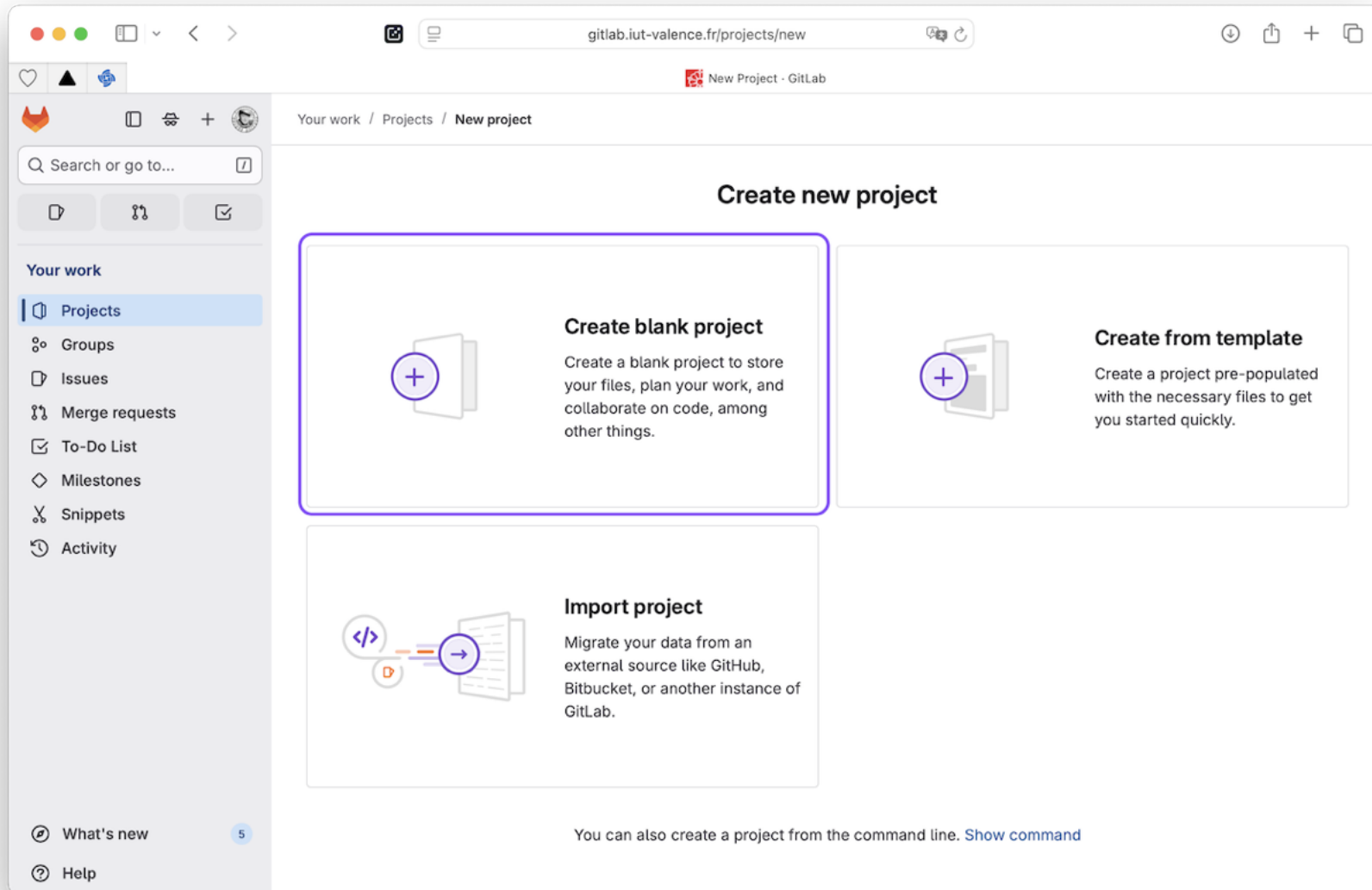
# Page d'accueil d'un compte Gitlab



# Page d'accueil d'un compte Gitlab (suite)



# Création d'un projet



# Créer un projet *From scratch*

gitlab.iut-valence.fr/projects/new#blank\_project

New Project · GitLab

Your work / Projects / New project / Create blank project

## Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Project name**

My project

Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

**Project URL**

https://gitlab.iut-valence.fr/

Pick a group or namespace

**Project slug**

my-awesome-project

**Visibility Level**

☒ Private  
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

☐ Internal  
The project can be accessed by any logged in user except external users.

☐ Public  
The project can be accessed without any authentication.

**Project Configuration**

☒ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more.](#)

☐ Enable Secret Detection  
Scan your code for secrets and credentials to prevent unauthorized access. [Learn more.](#)

Create project Cancel

# Créer un projet *From scratch*

The screenshot shows the 'Create blank project' page on GitLab. The browser address bar shows 'gitlab.iut-valence.fr/projects/new#blank\_project'. The page title is 'New Project · GitLab'. The breadcrumb trail is 'Your work / Projects / New project / Create blank project'.

**Create blank project**  
Create a blank project to store your files, plan your work, and collaborate on code, among other things.

**Project name**  
La classe américaine  
Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

**Project URL**  
https://gitlab.iut-valence.fr/ abitol

**Project slug**  
la-classe-americaine

**Visibility Level**  
☒ Private  
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.  
☐ Internal  
The project can be accessed by any logged in user except external users.  
☐ Public  
The project can be accessed without any authentication.

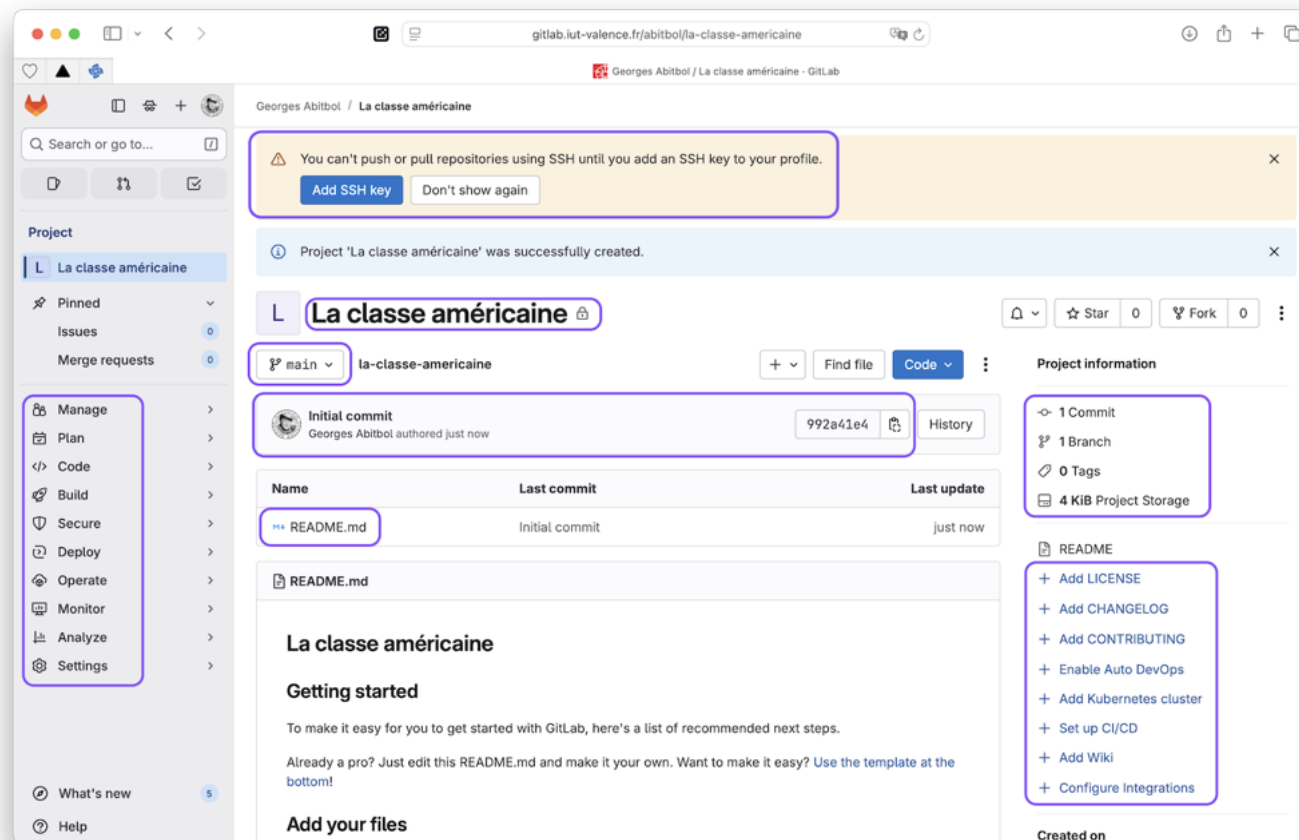
**Project Configuration**  
☒ Initialize repository with a README  
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.  
☐ Enable Static Application Security Testing (SAST)  
Analyze your source code for known security vulnerabilities. [Learn more.](#)  
☐ Enable Secret Detection  
Scan your code for secrets and credentials to prevent unauthorized access. [Learn more.](#)

**Annotations:**  
A dashed box highlights the README configuration section with the text: 'Le fichier README.md décrit le projet' and 'L'historique sera initialisé avec un premier commit contenant ce fichier'.

**Buttons:** 'Create project' and 'Cancel'.

# Créer un projet *From scratch* (suite)

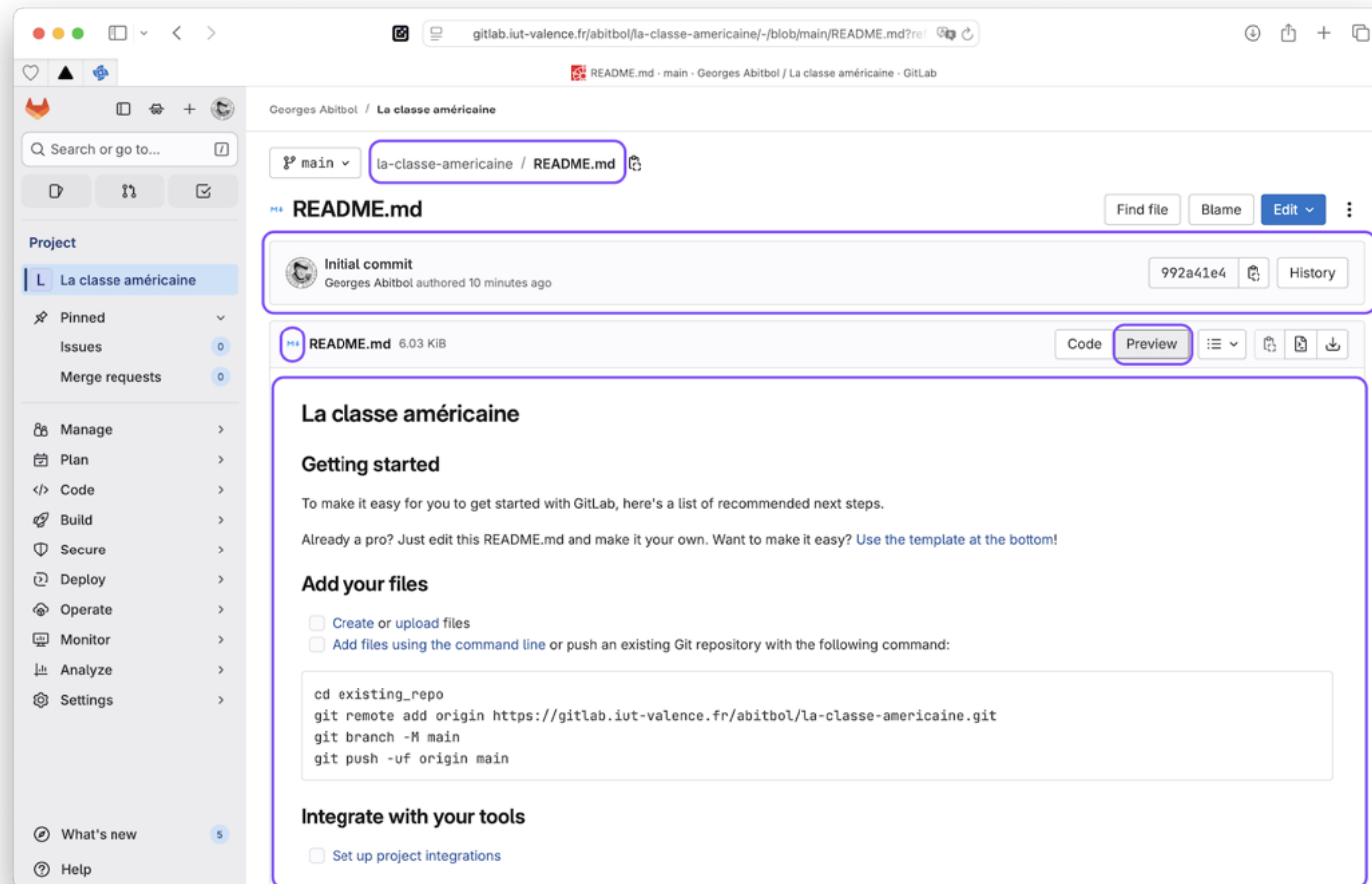
- **Projet** = **Dépôt** + services associés



- L'historique possède 1 commit, on peut visualiser le *Working Tree*

# Le fichier README.md

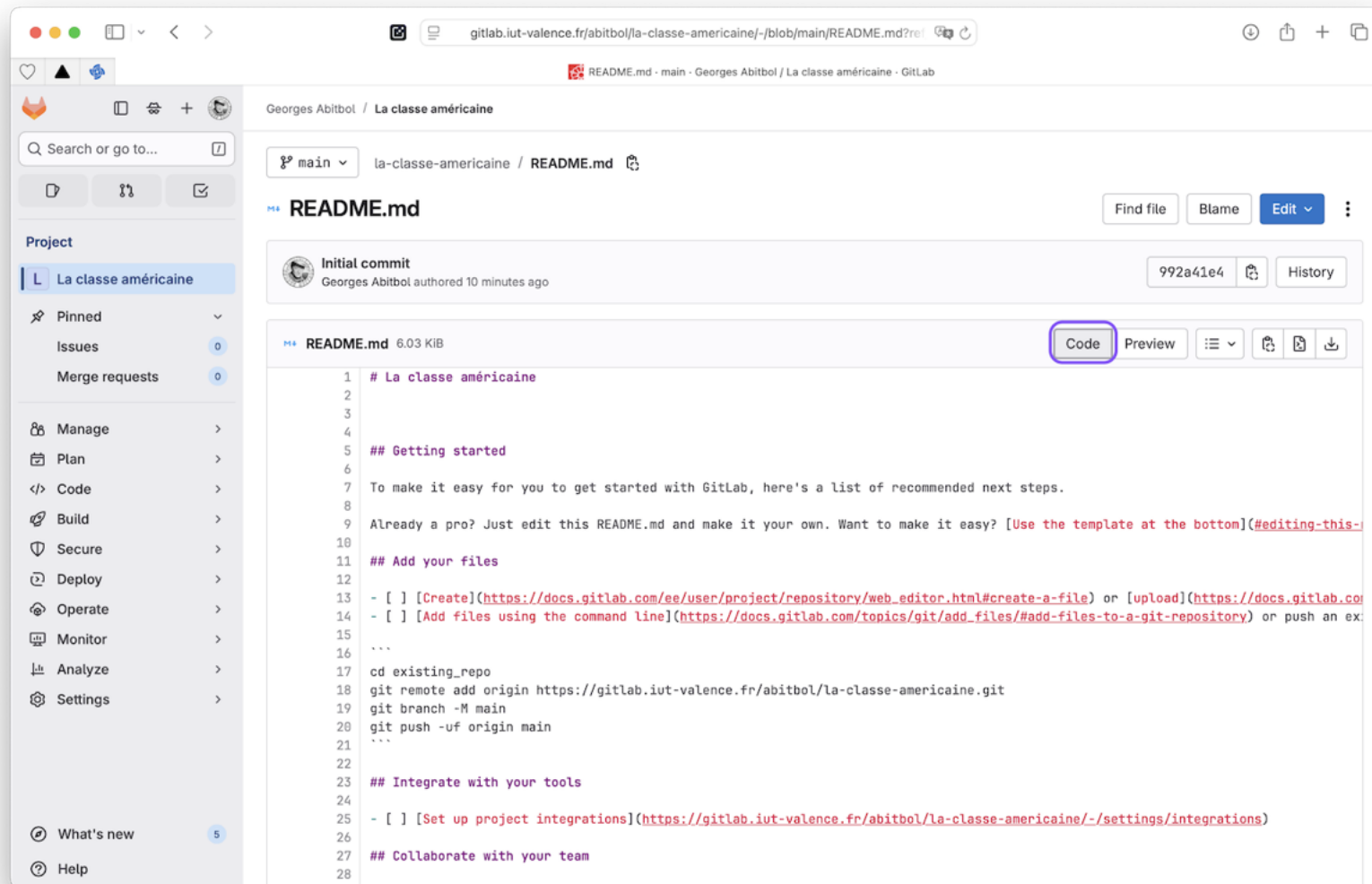
- Le contenu de certains fichiers est **prévisualisé** quand c'est possible



- Le fichier README.md est un fichier texte au format **Markdown**, il est *rendu* en HTML

# Le fichier README.md (suite)

- Il est toujours possible de voir le contenu *brut* d'un fichier texte



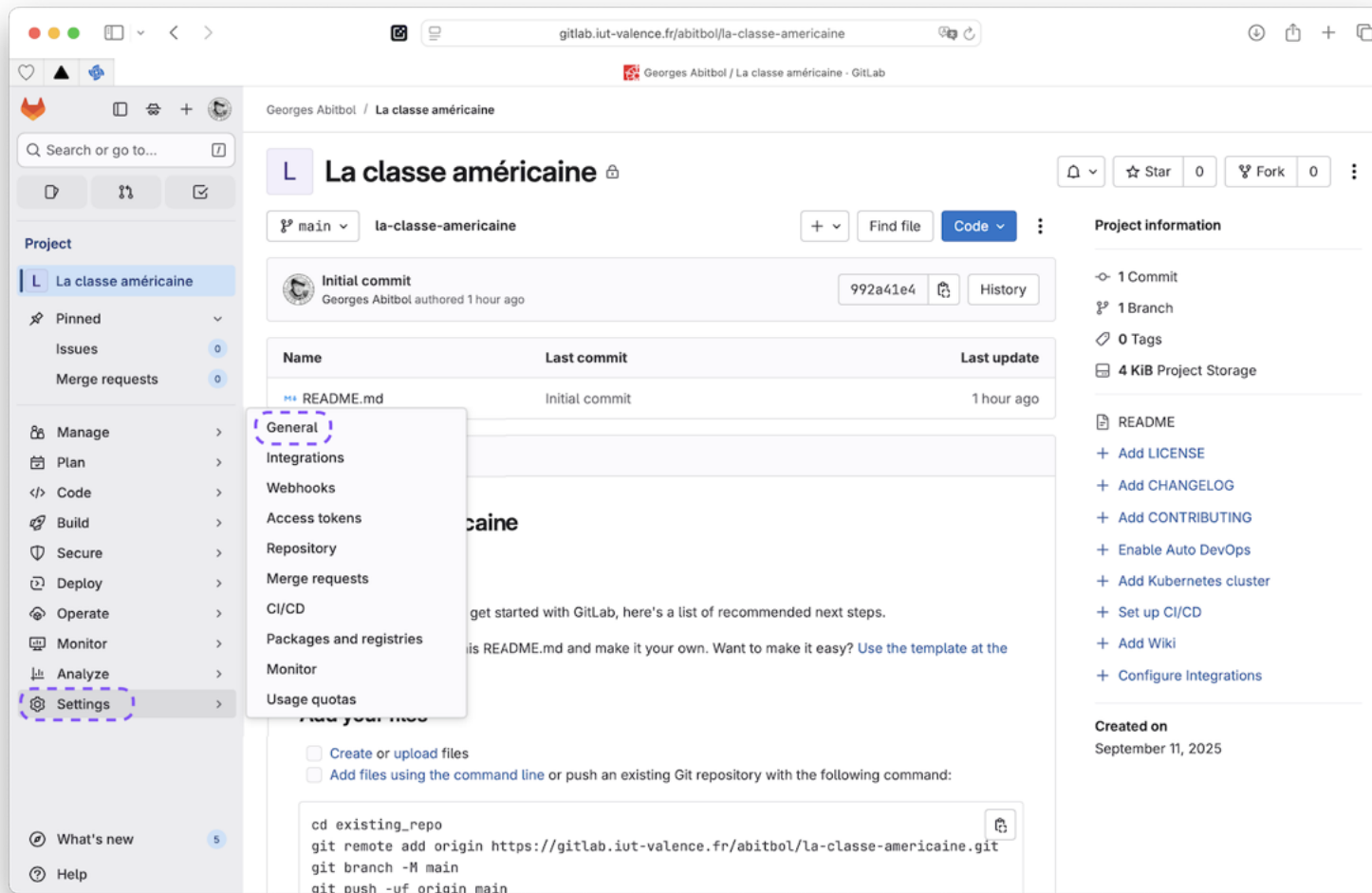
# Interlude : *MarkDown*



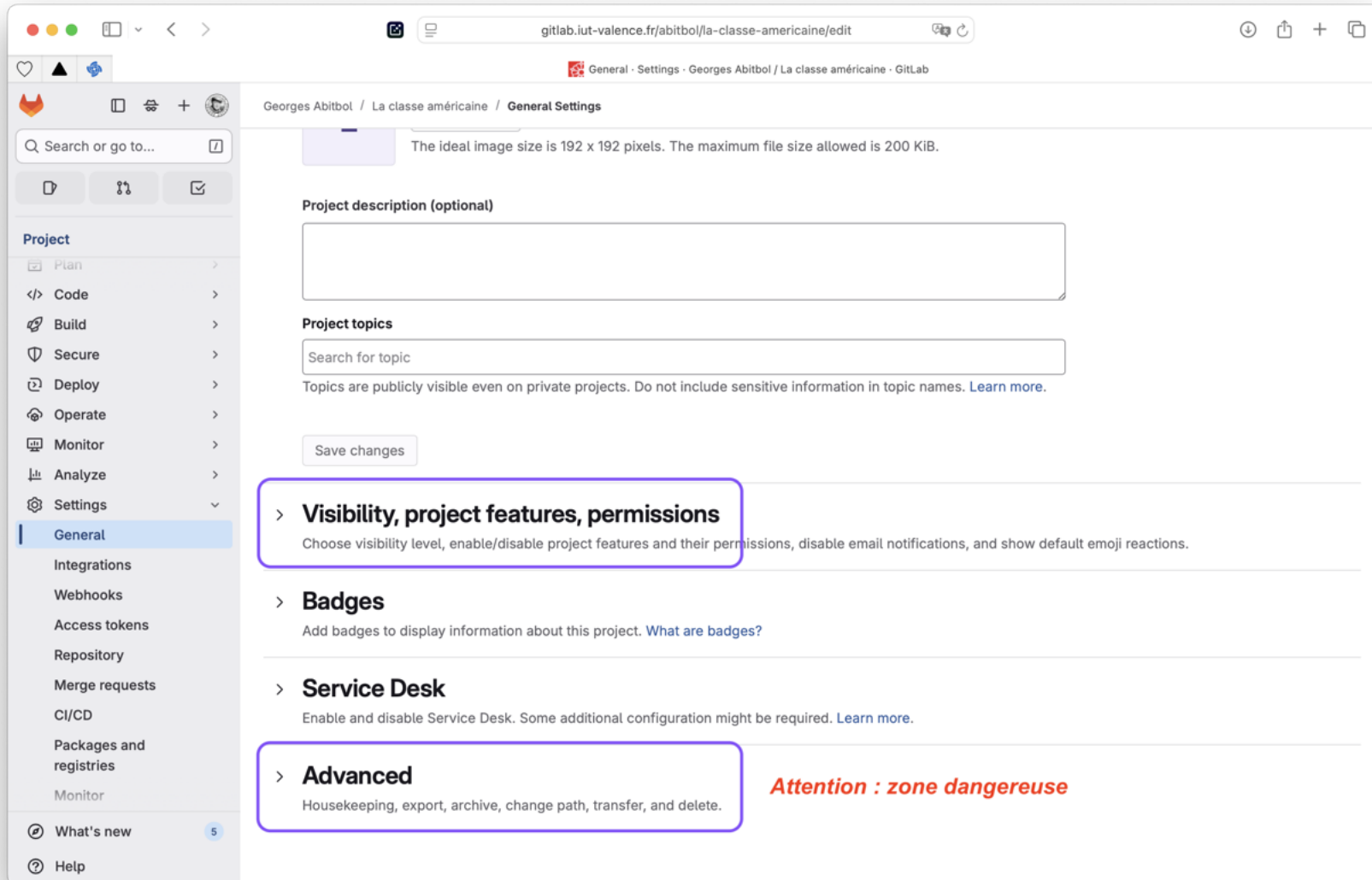
- **Trouver** le *MarkDown Cheat Sheet*
- **Retrouver les éléments du langage** en faisant des aller-retours entre Code et Preview
- **Expérimenter** 5 minutes avec <https://stackedit.io/app#>

# Configuration d'un projet

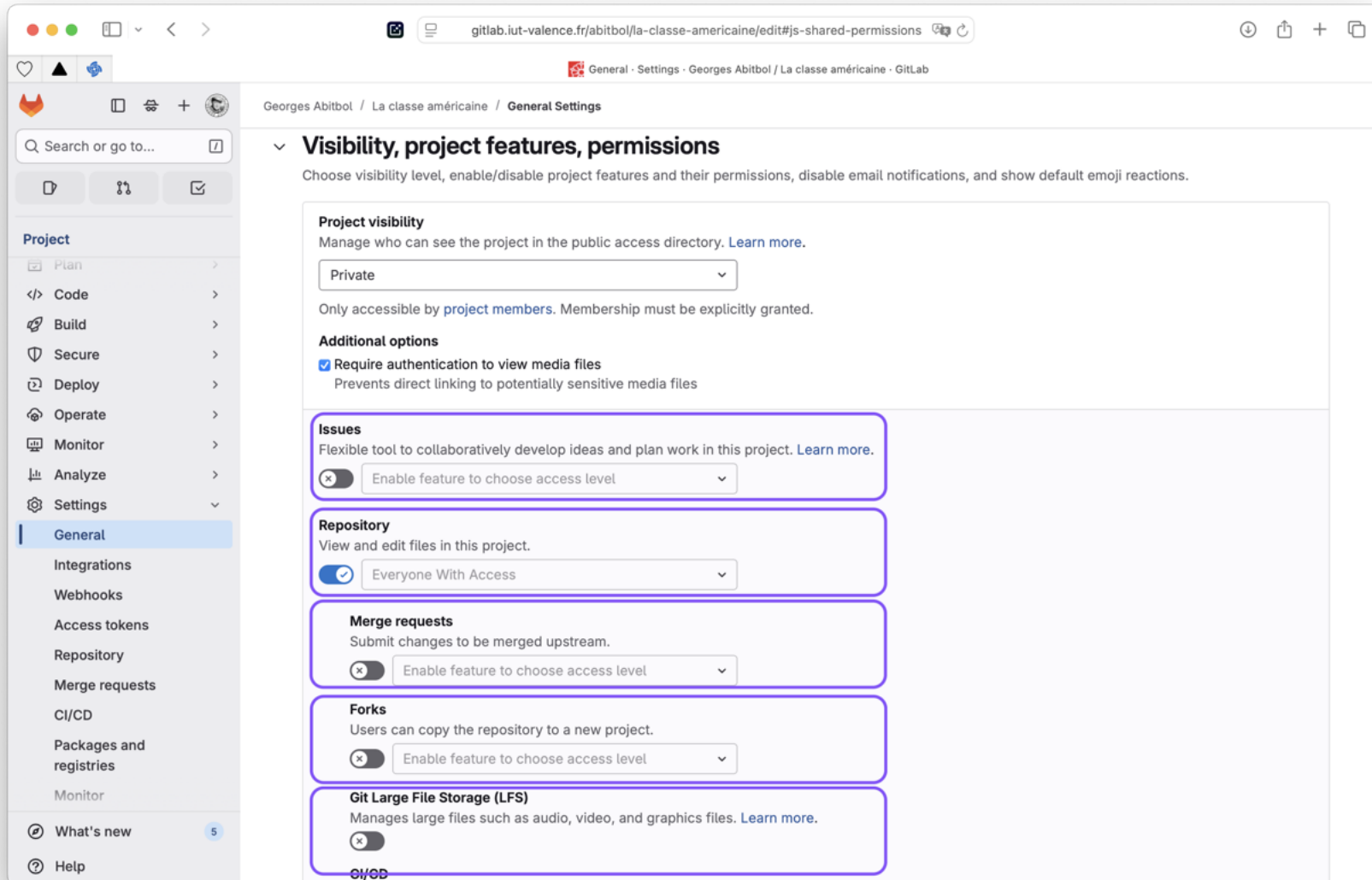
- N.B. : pour de la **simple gestion de versions**, seules les rubriques Code et Manage sont utiles



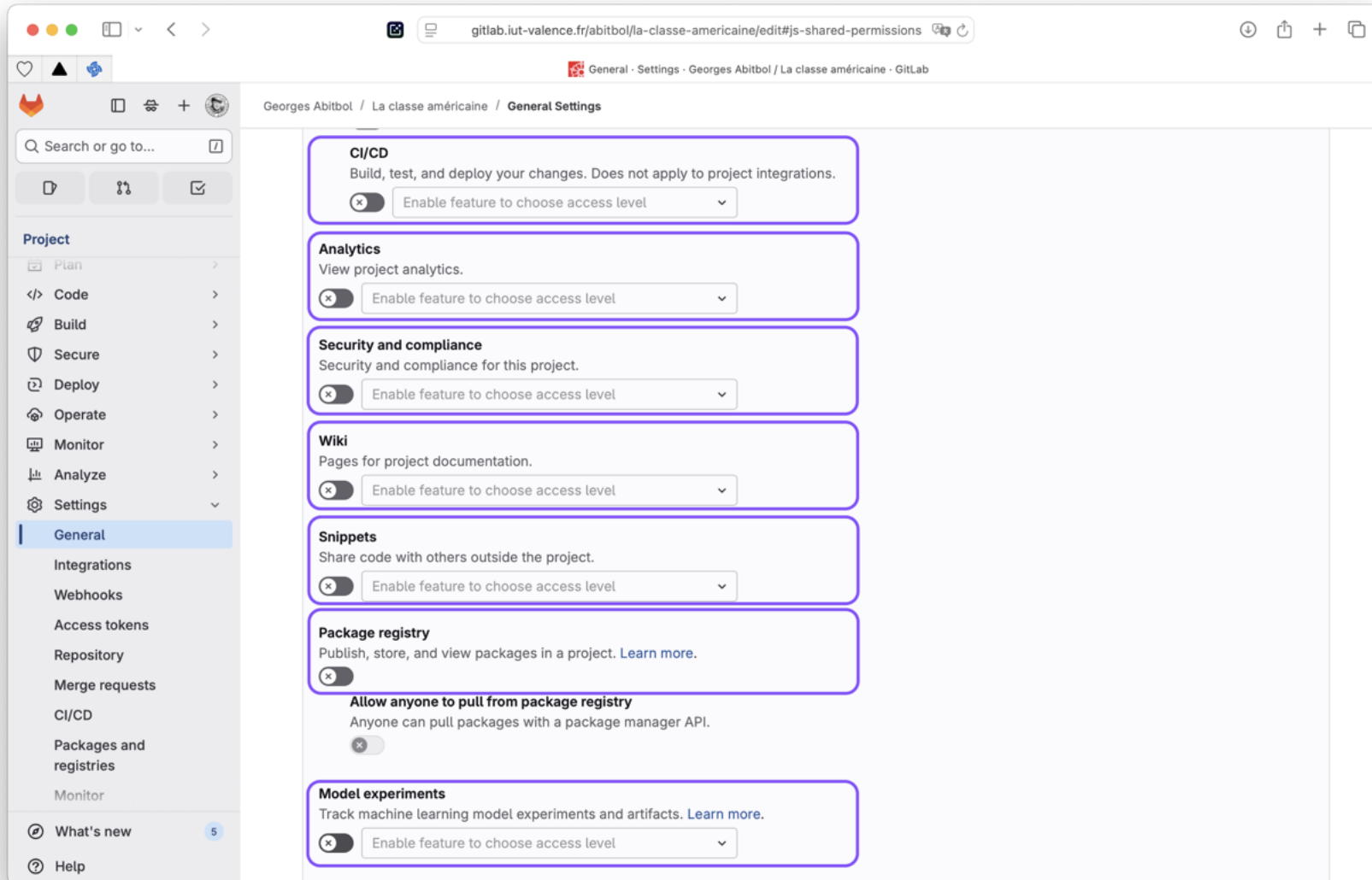
# Configuration d'un projet



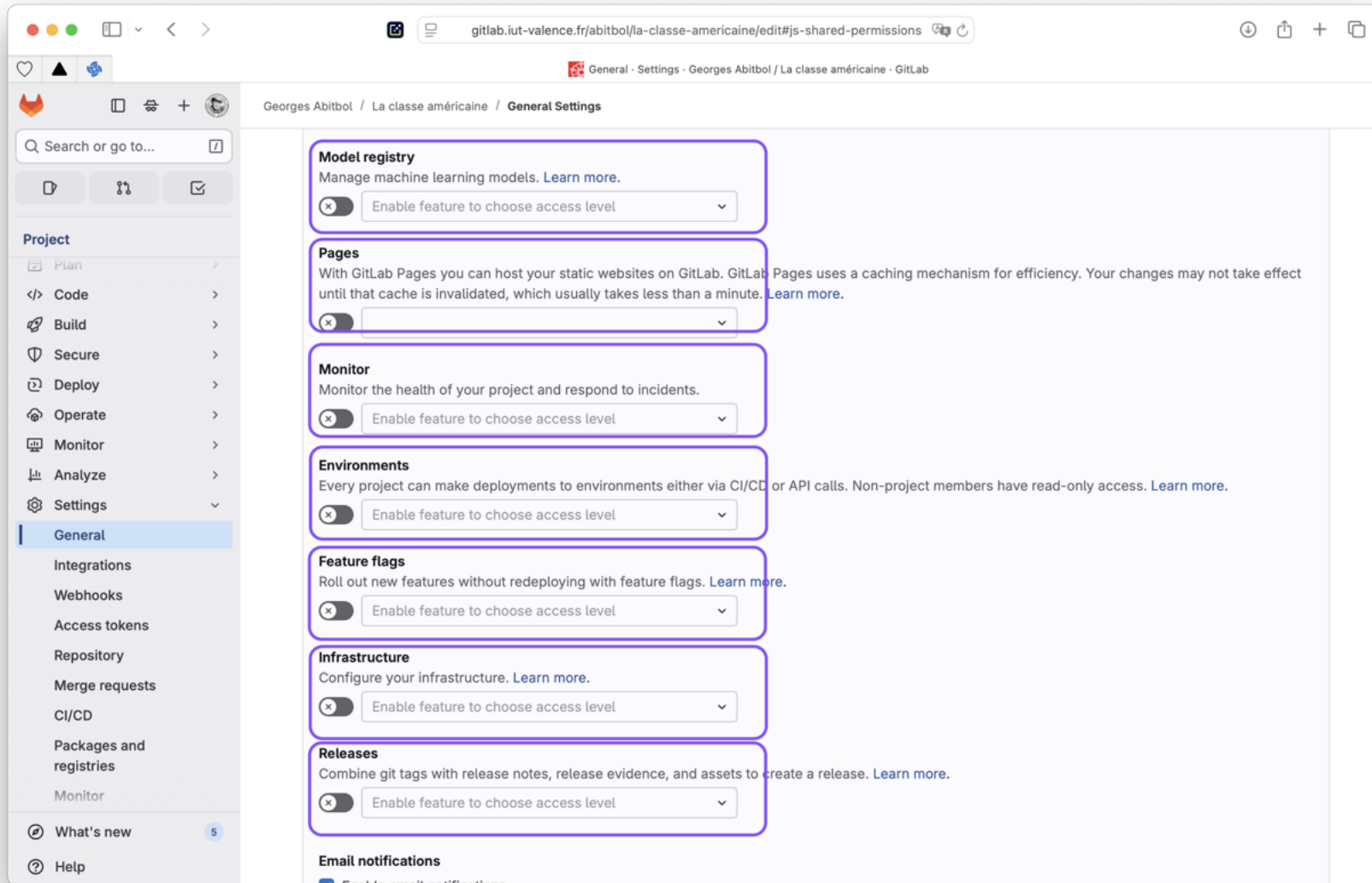
# Réduire l'interface au minimum vital



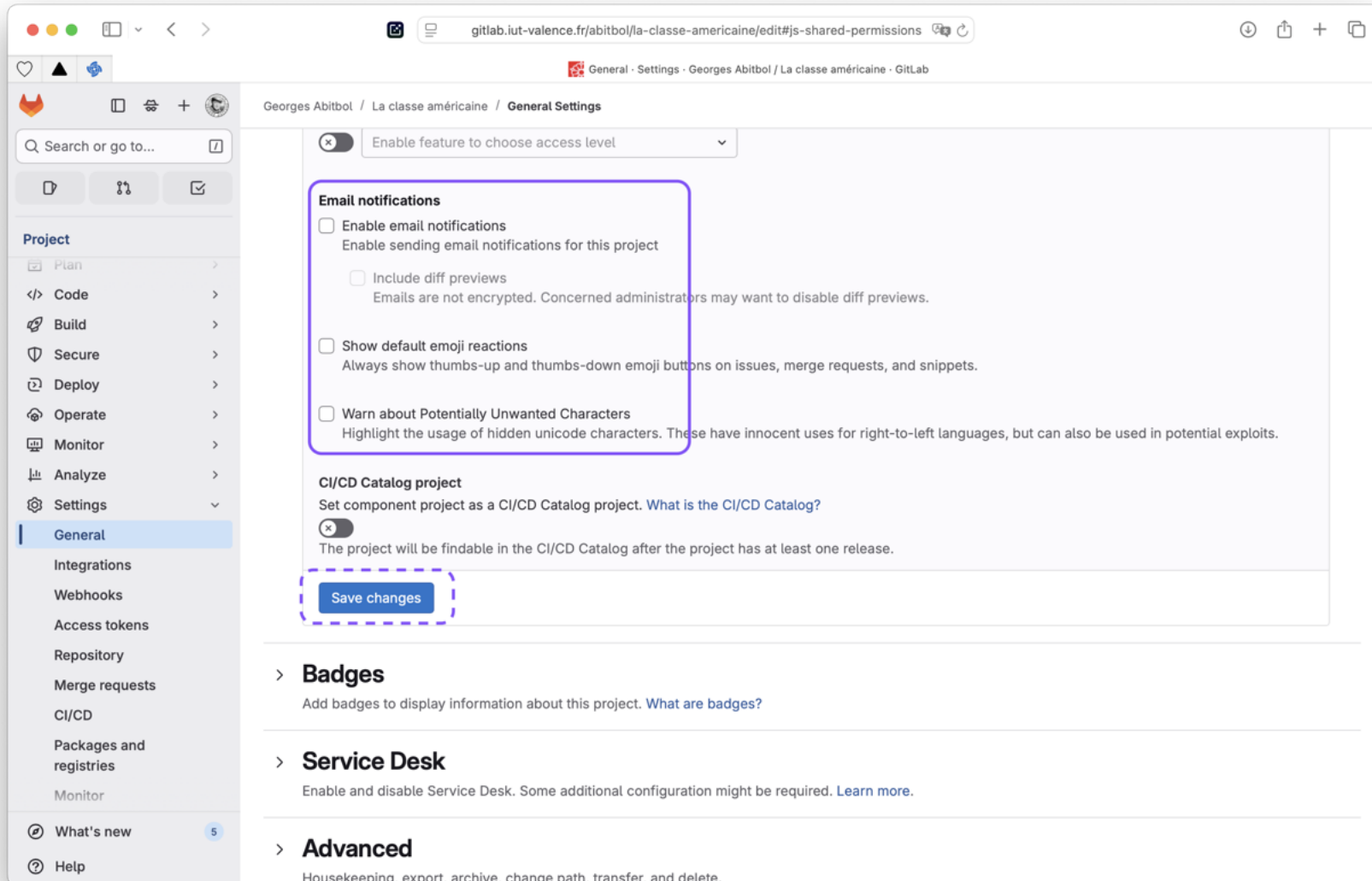
# Réduire l'interface au minimum vital (suite)



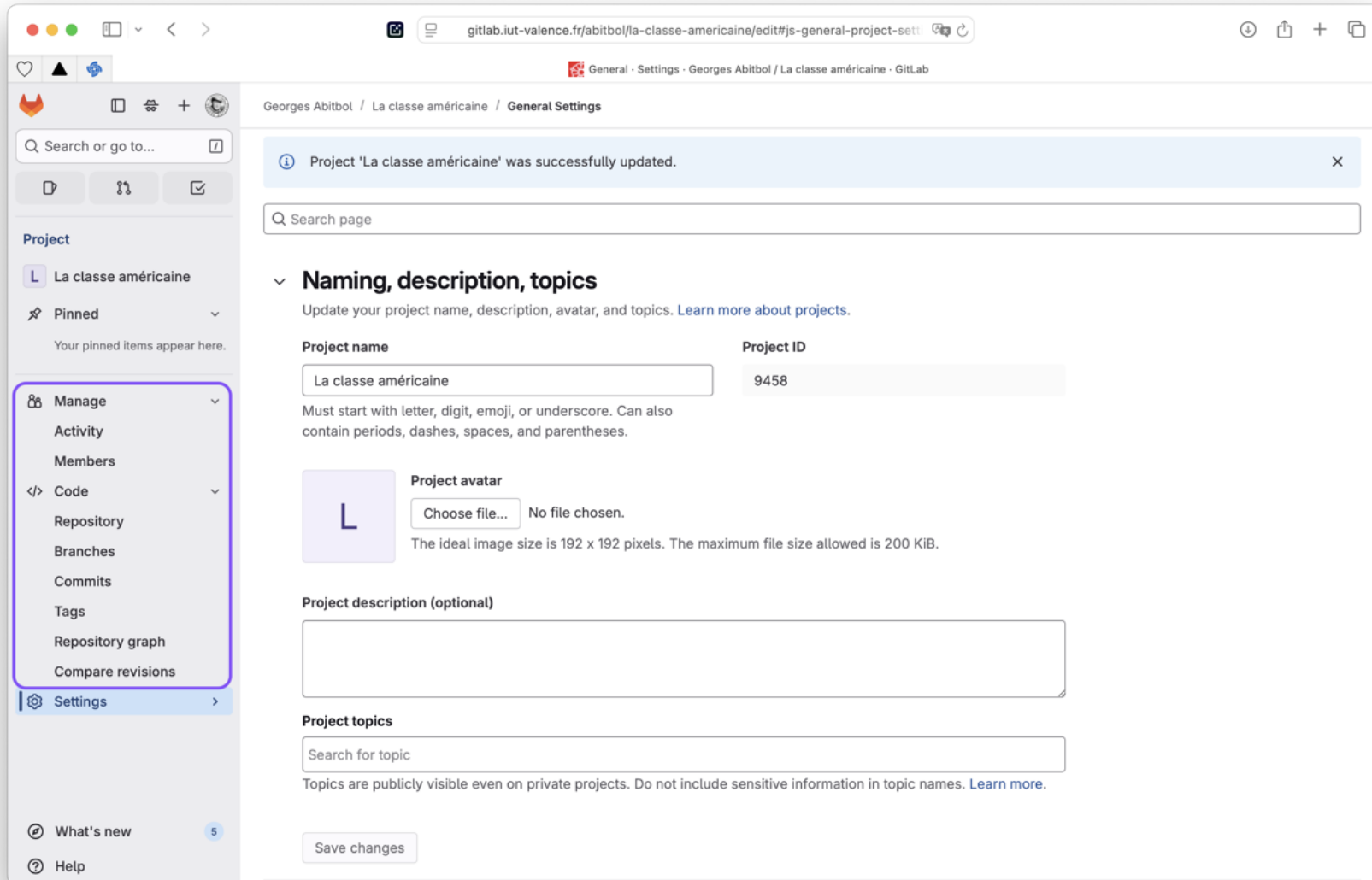
# Réduire l'interface au minimum vital (suite)



# Réduire l'interface au minimum vital (suite)

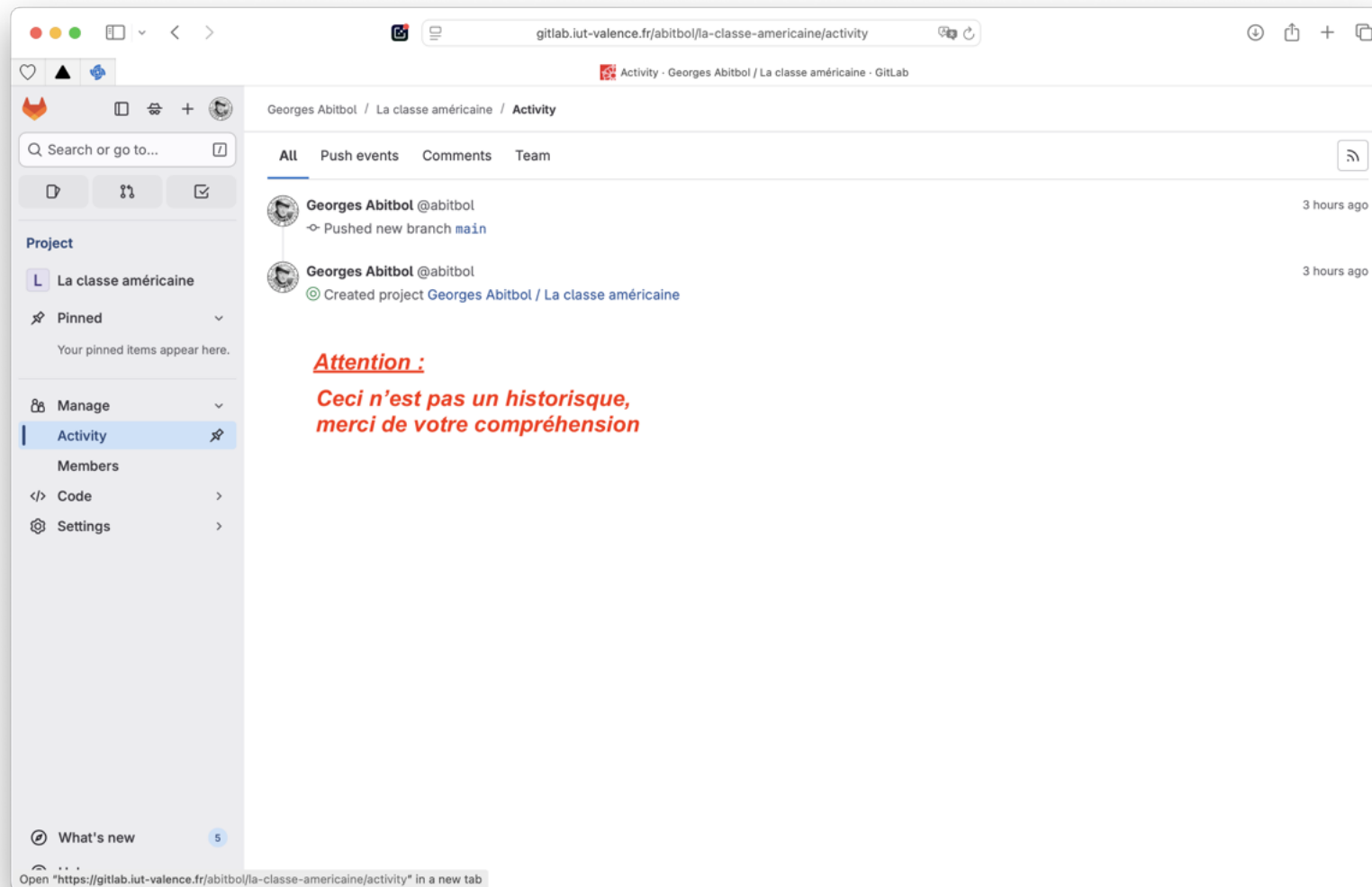


# Réduire l'interface au minimum vital (fin)



# Visualiser l'activité sur un projet

- **Vie du projet** (*qui a fait quoi quand*)  $\neq$  Historique du dépôt



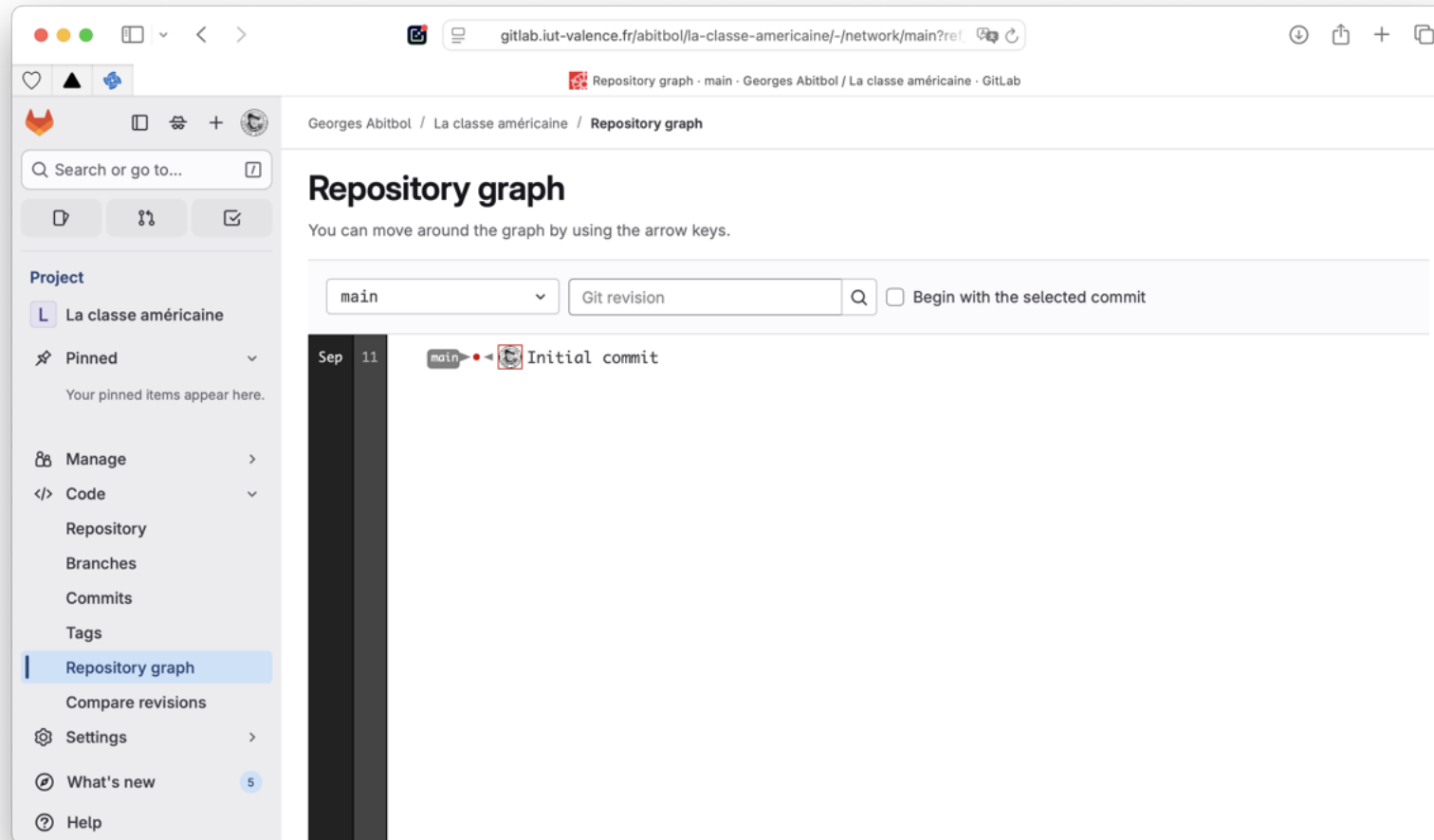
# Visualiser les membres associés au projet

- Différents niveaux de droits (propriétaire → ... → invité)

The screenshot shows the GitLab web interface for the project 'La classe américaine'. The left sidebar contains navigation links: Project, Pinned, Manage, Activity, Members (selected), Code, Settings, What's new, and Help. The main content area is titled 'Project members' and includes buttons for 'Import from a project', 'Invite a group', and 'Invite members'. Below this, a table lists the project members. The table has columns for Account, Source, Role, Expiration, and Activity. One member is listed: Georges Abitbol (@abitbol), who is the Owner of the project. The role 'Owner' is highlighted with a purple circle. The activity section shows dates for 8+ Sep 11, 2025, Sep 11, 2025, and Sep 11, 2025.

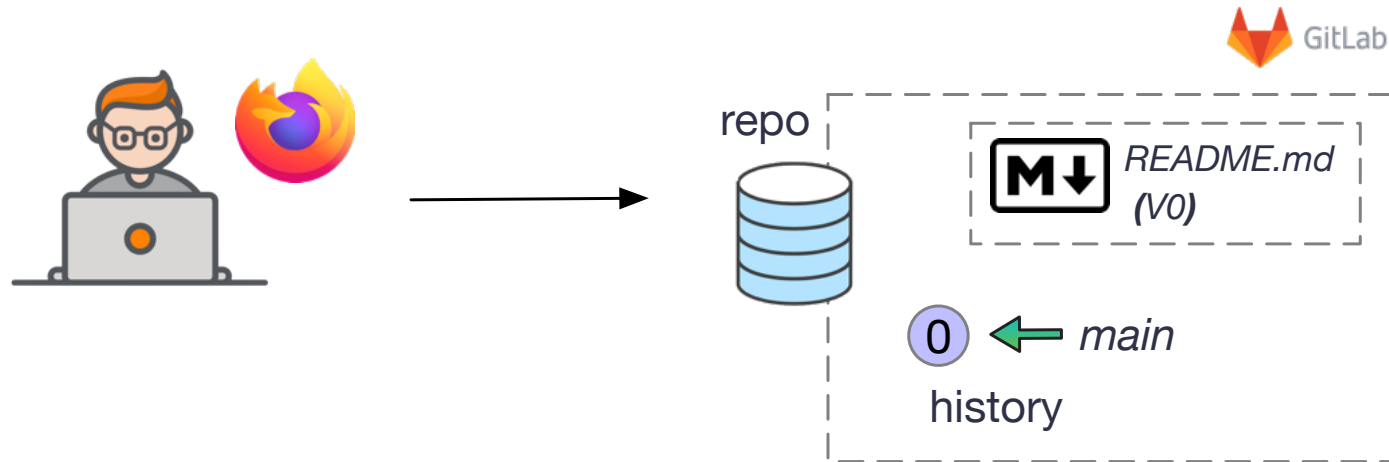
| Account                                                                                                                             | Source                              | Role  | Expiration                                                                                          | Activity                                            |
|-------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-------|-----------------------------------------------------------------------------------------------------|-----------------------------------------------------|
|  Georges Abitbol <span>It's you</span><br>@abitbol | Direct member by<br>Georges Abitbol | Owner | Expiration date  | 8+ Sep 11, 2025<br>✓ Sep 11, 2025<br>⌚ Sep 11, 2025 |

# Visualiser l'historique du dépôt



- La vue *Repository* (déjà vue) montre le *Working Tree*
- La vue *Commits* montre les commits par ordre chronologique
- La vue *Compare revisions* fait l'équivalent de `git diff`

# Prise en main de Gitlab : rembobinage



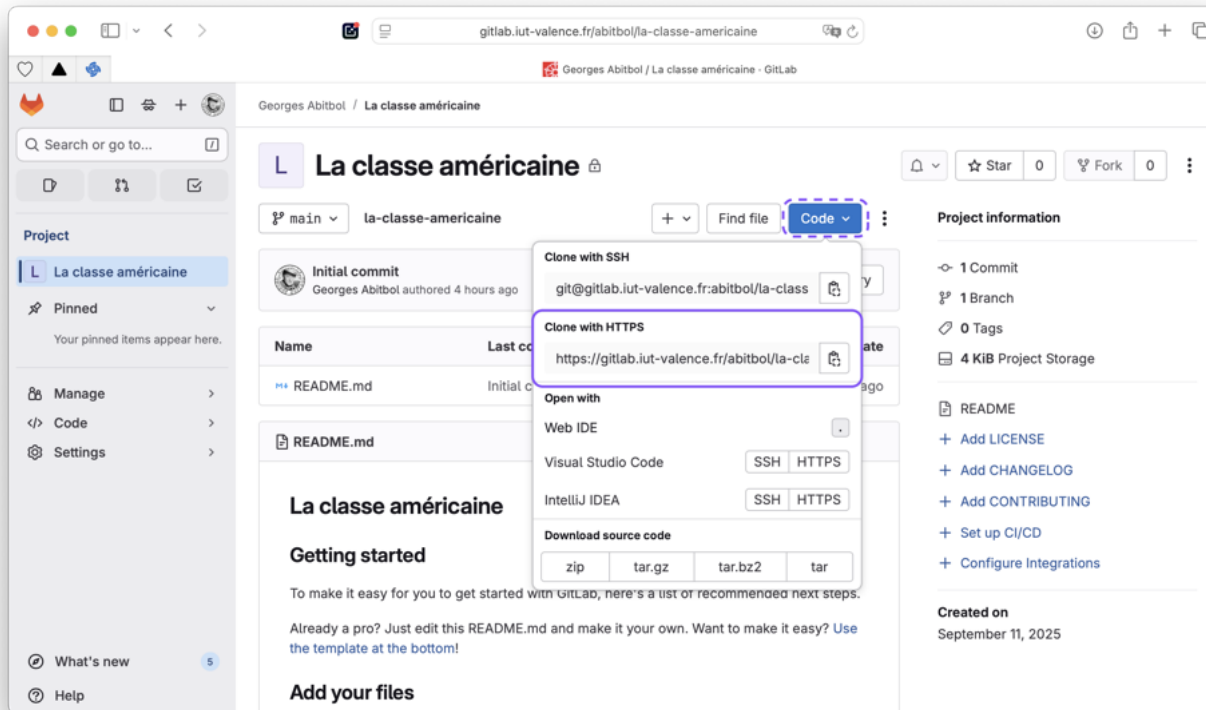
- Depuis l'interface Web d'un compte sur un serveur Gitlab on a créé un **projet** qui héberge un dépôt et des services associés
  - Ce dépôt a été **initialisé** avec un fichier README.md (qui est censé décrire à quoi sert ce dépôt) qui constitue le **premier commit**
  - Un dépôt distant est dit **bare**, il ne contient pas de *working tree* mais **simplement l'historique**
    - La référence HEAD n'a pas de sens, elle n'existe pas

# Travailler localement sur du code hébergé à distance

- Si l'on veut **contribuer au développement sur le dépôt distant**, on ne le fait **pas via l'interface web** mais localement sur son poste de travail (cf. TP précédent)
- On doit donc d'abord **reconstruire localement** un dépôt local **clone** du dépôt distant
  - L'**historique** est reproduit intégralement
  - Le *working tree* est reconstruit en rejouant intégralement l'historique
  - Le **clone local** et son **dépôt origine** restent **liés**



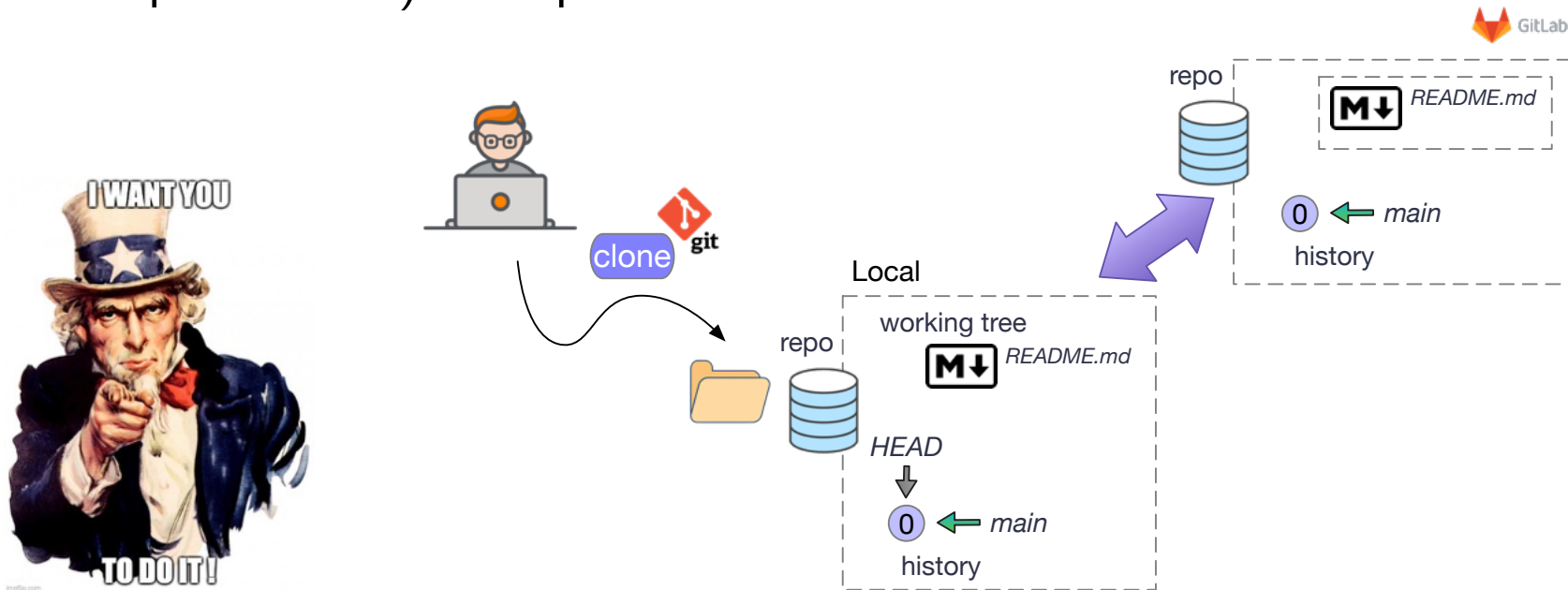
# Interaction avec un dépôt distant



- Un **dépôt distant** est **identifié par un URL**
  - **protocole + serveur + chemin**
- La **communication client/serveur** (authenticée / chiffrée) entre le client (local) et le serveur (distant) utilise soit **HTTPS** soit **SSH**
  - On laisse SSH pour plus tard ...

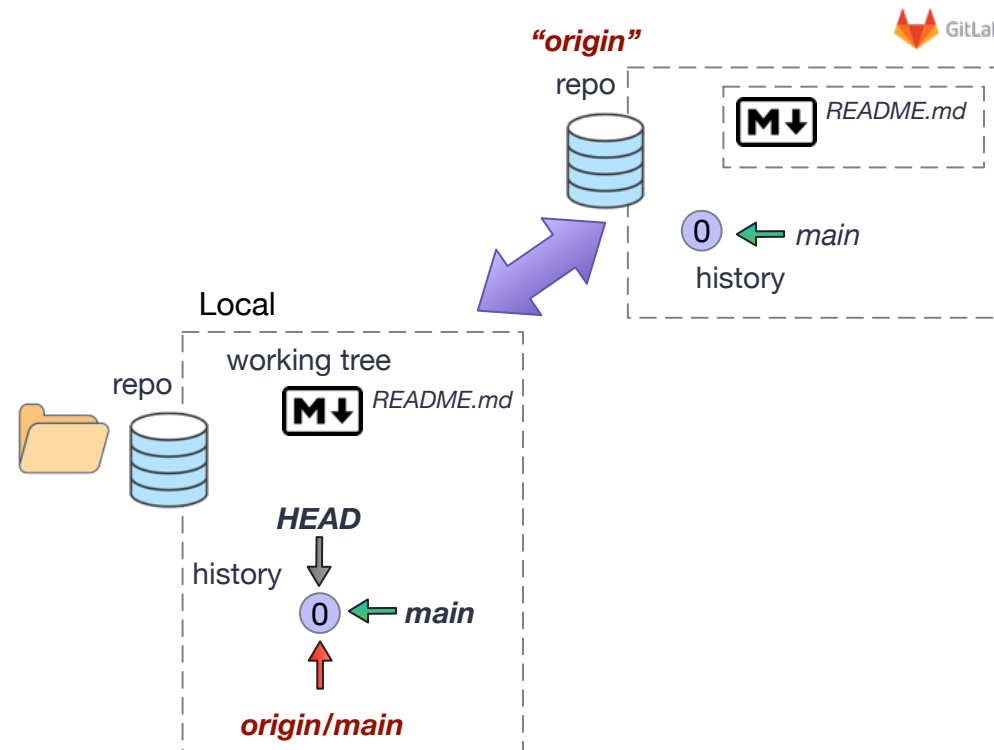
# Clonage

- Le **clonage** est effectué via la commande **git clone (+ URL)**
  - Le dépôt distant sera cloné dans un sous répertoire (portant le nom du dépôt distant) du répertoire où cette commande est exécutée



- **Cloner le dépôt distant** à la **racine du compte utilisateur local**
- **Observer l'historique local et l'historique distant**
- **Observer** le fichier local contenant la **configuration du dépôt**

# Clonage



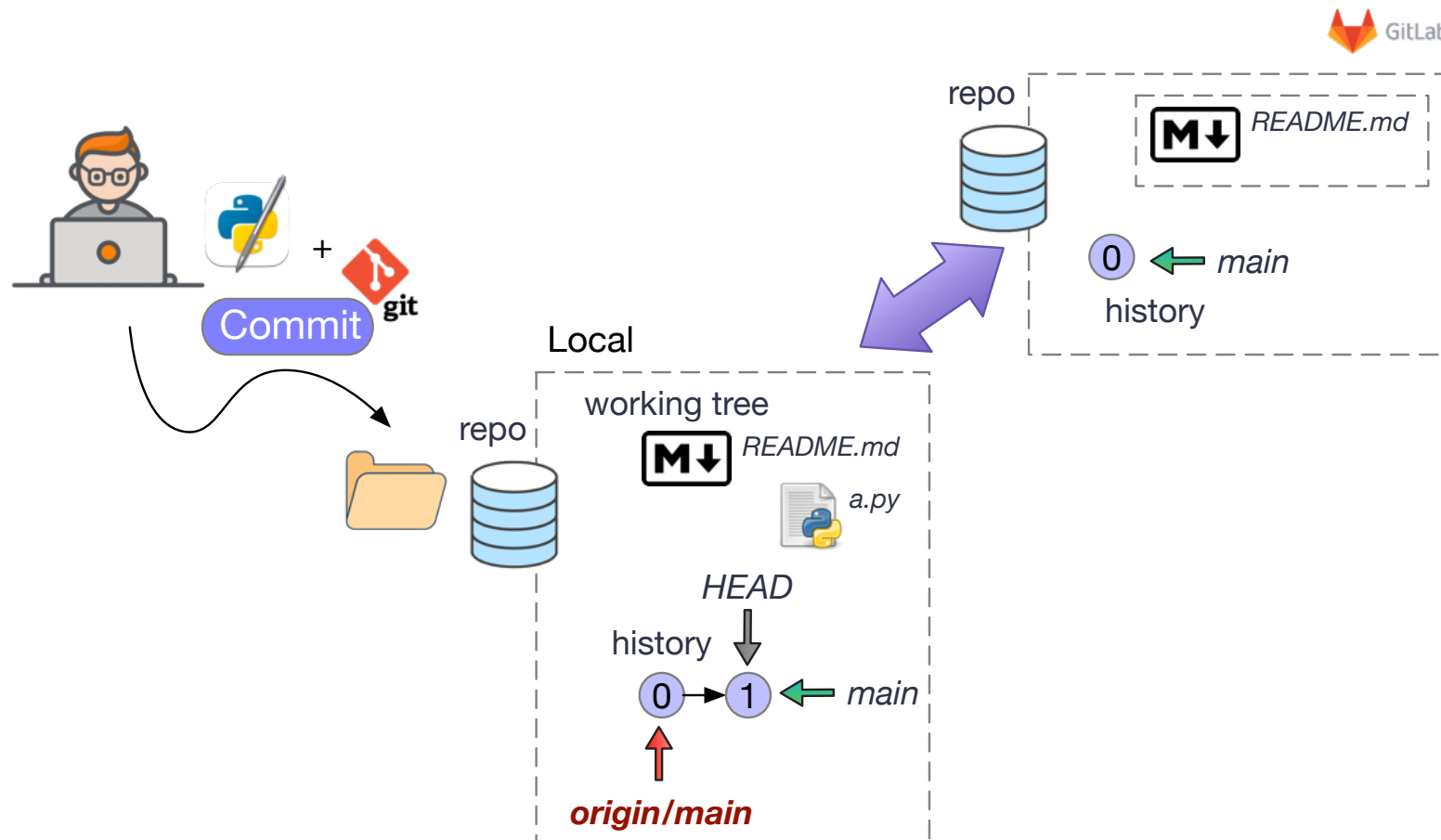
- Le clone et son dépôt original (*origin*) restent liés
- Le **marqueur origin/main** repère le **commit distant le plus récent connu localement**
  - Quand main et origin/main désignent le même commit alors on pense localement avoir le même contenu qu'à distance

# Produire une nouvelle version localement



- Créer un fichier `a.py` avec un contenu non vide
- Effectuer un **commit** des modifications (ajout du fichier)
- Observer l'historique local et l'historique distant

# Commit = local



- git commit est une **opération toujours locale**, sans communication avec le serveur
  - L'historique local change, **l'historique distant ne change pas**

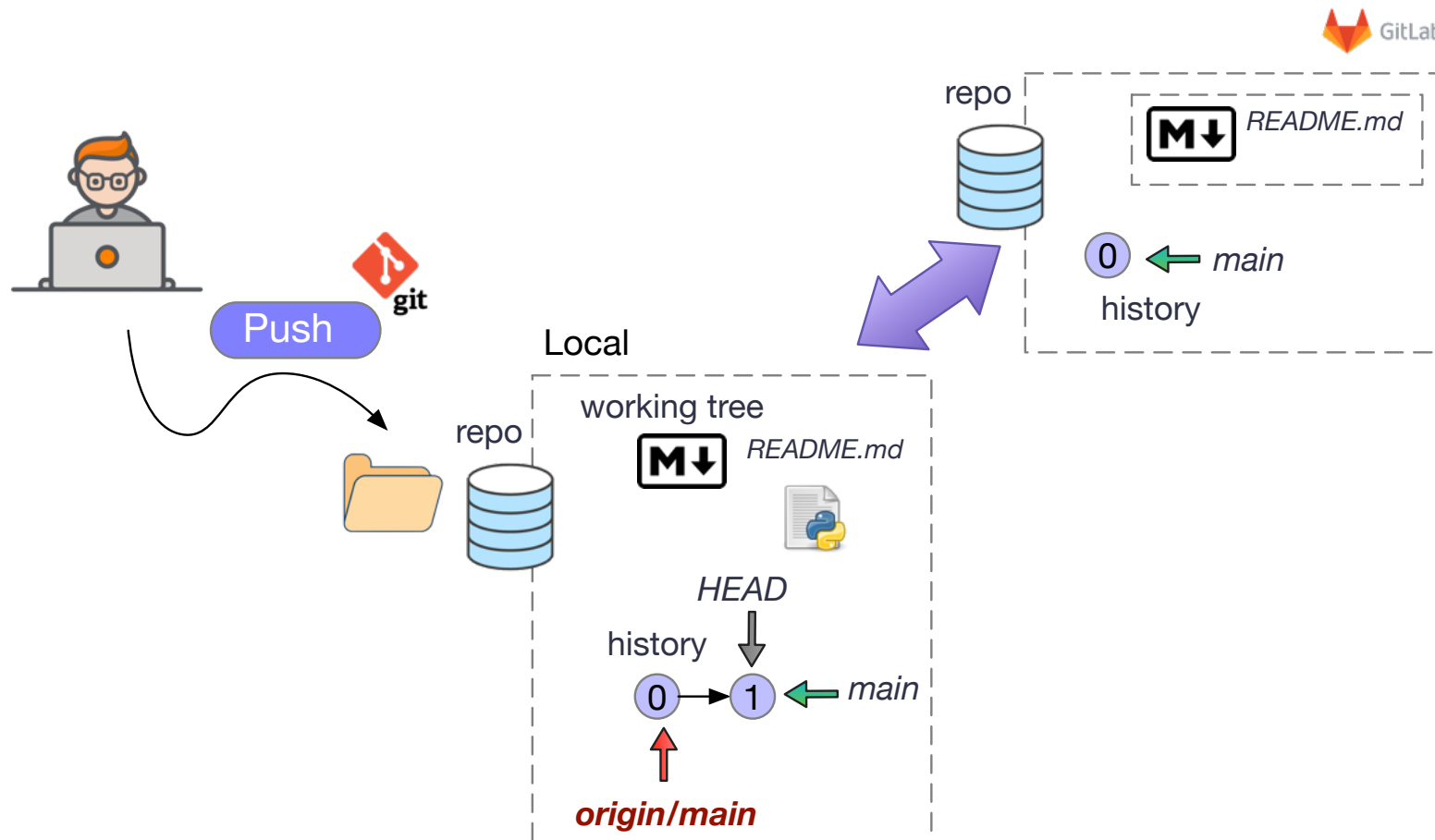
# Mise à jour du dépôt distant à partir du local



- **Intégrer à l'historique distant** les **commits locaux manquants** avec la commande `git push`
- **Observer** l'historique local et l'historique distant

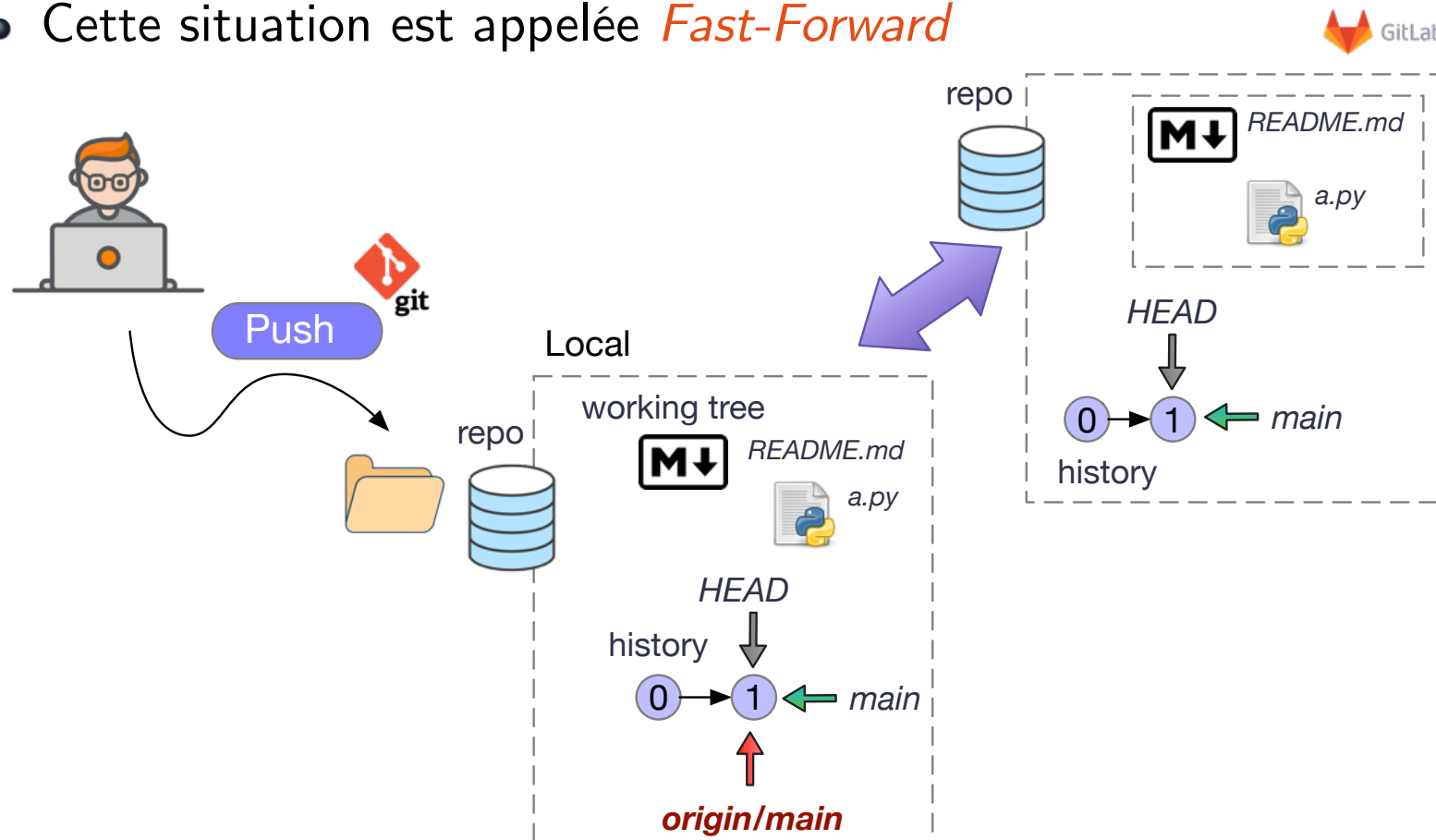
# Mise à jour du dépôt distant à partir du local

- L'exécution de la commande `git push` commence par identifier localement les commits qui sont absents à distance
  - Par comparaison des positions des marqueurs `main` et `origin/main`

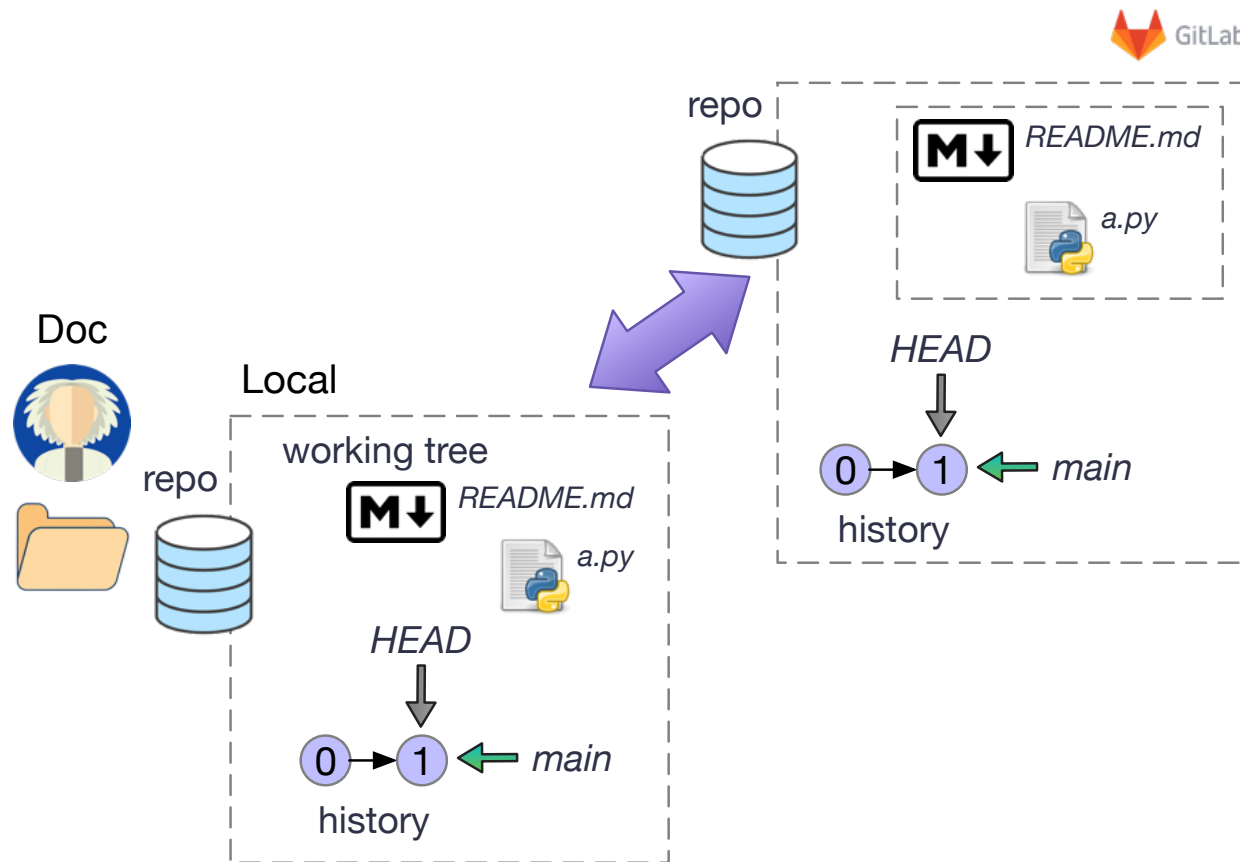


# Mise à jour du dépôt distant à partir du local

- Le **client** émet ensuite une **requête contenant les commits manquants** pour que le serveur les intègre
  - Le serveur étudie la demande et l'**accepte** car il peut effectivement raccrocher 1 à 0 car 0 n'a pas de successeur à distance
  - Cette situation est appelée **Fast-Forward**

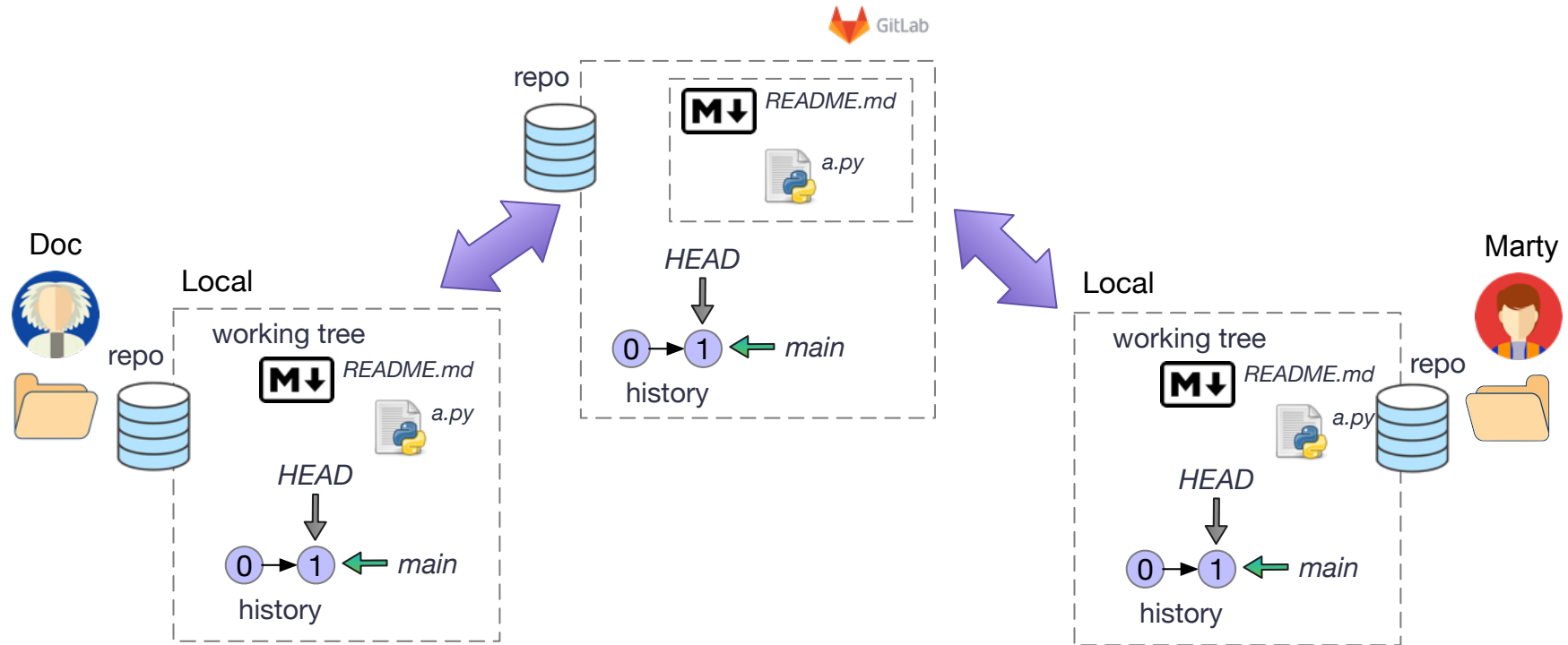


# Se remettre à jour avec le dépôt distant



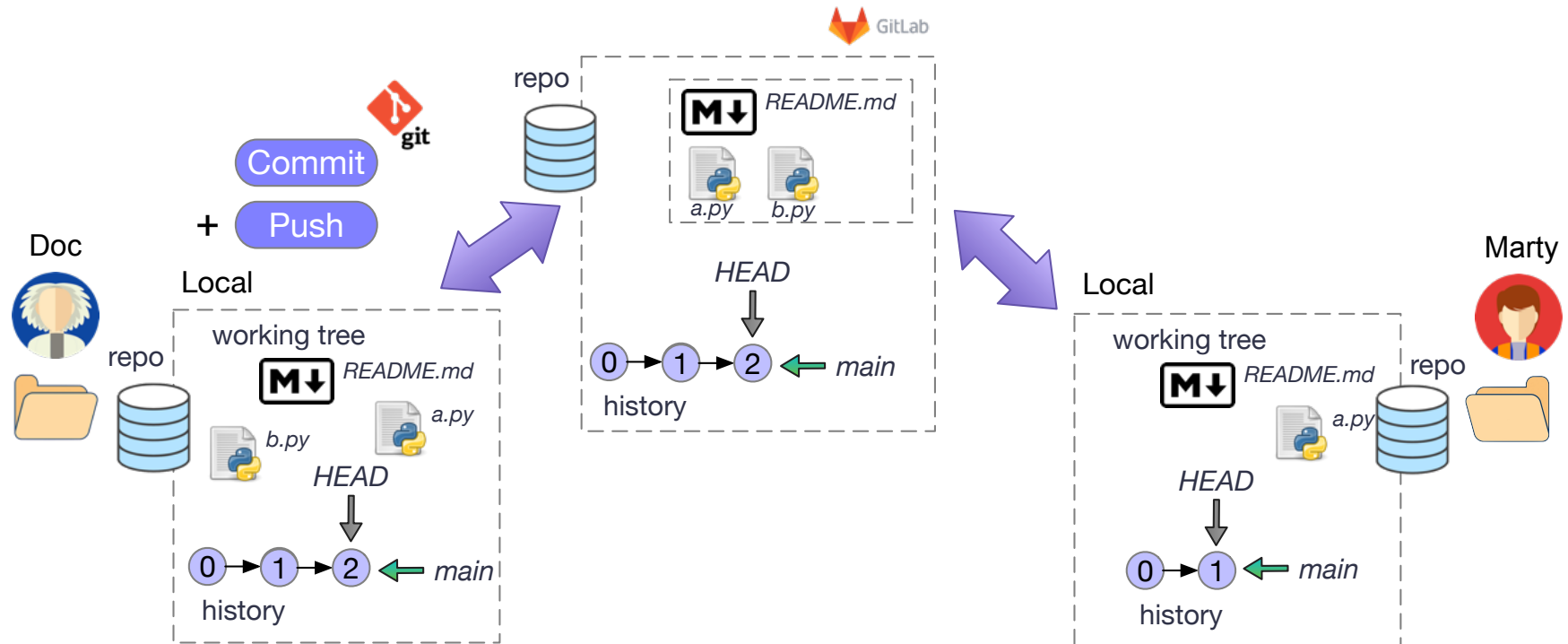
- *Doc* a créé un dépôt sur *gitlab*, l'a cloné localement (**clone**), a produit 1 nouvelle version localement (**commit**), l'a envoyée sur le serveur (**push**)

# Se remettre à jour avec le dépôt distant



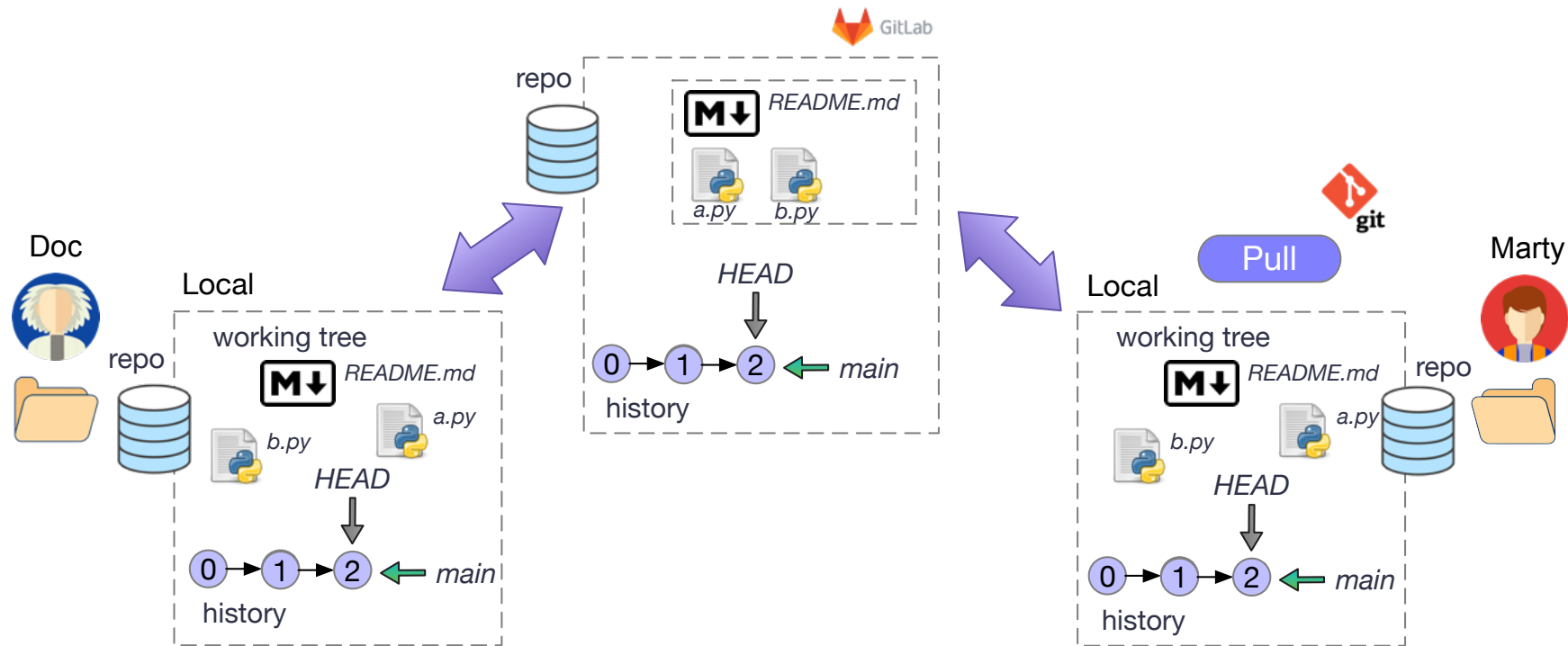
- *Marty* rejoint le développement, et clone à son tour le dépôt

# Se remettre à jour avec le dépôt distant



- *Doc* produit une nouvelle version locale qu'il envoie au serveur
- *Marty* n'est pas averti de l'existence de cette nouvelle version

# Se remettre à jour avec le dépôt distant



- Pour récupérer les éventuels commits qui lui manquent, *Marty* doit explicitement se synchroniser avec le serveur (**pull**)
- N.B. *ici, c'est un cas simple (fast-forward), qui se résout automatiquement et de manière autonome*
  - *Quelquefois, des situations de conflits apparaissent et nécessitent une intervention du développeur*

# Ajouter un contributeur sur le projet

The screenshot shows the GitLab interface for the project 'La classe américaine' by Georges Abitbol. The 'Project members' page is displayed, showing a list of members. The 'Invite members' button is highlighted in the top right. In the left sidebar, the 'Members' option under the 'Project' section is highlighted.

**Project members**

You can invite a new member to La classe américaine or invite another group.

**Members 1**

| Account                                                                                                                                    | Source                              | Role  | Expiration                                                                                          | Activity                                            |
|--------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------|-------|-----------------------------------------------------------------------------------------------------|-----------------------------------------------------|
|  <b>Georges Abitbol</b> <span>It's you</span><br>@abitbol | Direct member by<br>Georges Abitbol | Owner | Expiration date  | 8+ Sep 11, 2025<br>✓ Sep 11, 2025<br>✗ Sep 11, 2025 |

# Pour finir ...



- **Ajouter un membre** en tant de **développeur** sur le dépôt
- **Expérimenter** en jouant le rôle du Doc (*commit / push*) pendant que l'autre membre le rôle de Marty (*clone / pull*) ...
- ...en **observant** à chaque étape **les historiques locaux et distants**

# S'il reste du temps

- **Aller observer** un **projet conséquent** sur `gitlab.com`
  - Par exemple  
`https://gitlab.com/swiss-armed-forces/cyber-command/cea/loom`
- **Créer un compte** sur `github.com`, **créer un projet** et **comparer** les **interfaces** de GitHub et Gitlab

# Fin !

