

# Gestion de versions avec Git

Xavier Gendre [in](#)

## Système de gestion de version

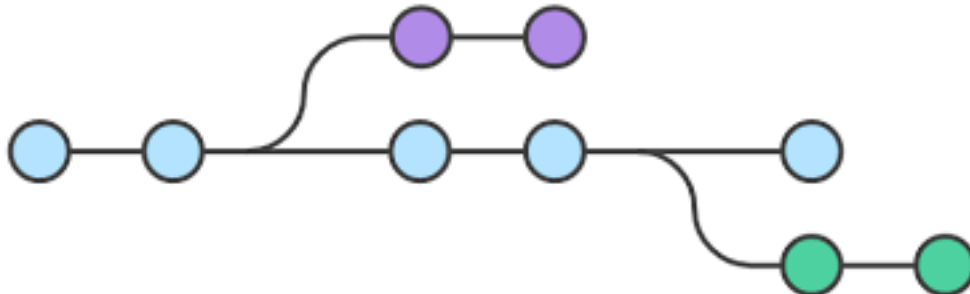
Objectifs principaux :

- Travailler à plusieurs
  - Partage du code
  - Gestion des modifications
  - Mutualisation
- Historique des changements
  - Retour en arrière
  - Différences entre 2 versions d'un fichier
  - Qui a modifié un fichier en dernier ?

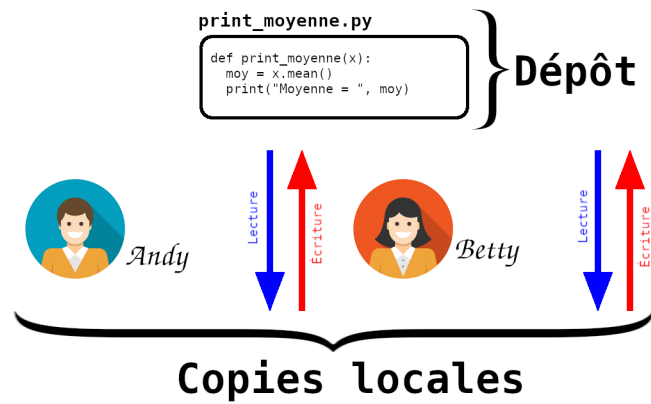
---

Autres objectifs de la gestion de version :

- Gestion des **branches** : mener en parallèle plusieurs versions (stable, dev, ...)
- Utilisation de **tags** : donner un nom explicite à une version pour y accéder facilement
- Sécurité : intégrité, confidentialité, ...

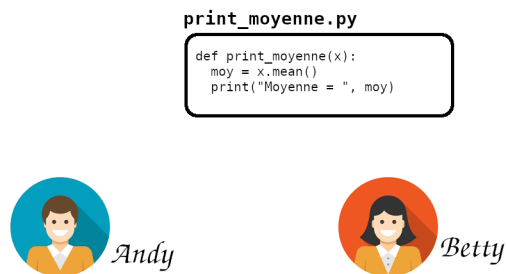


## Principe général



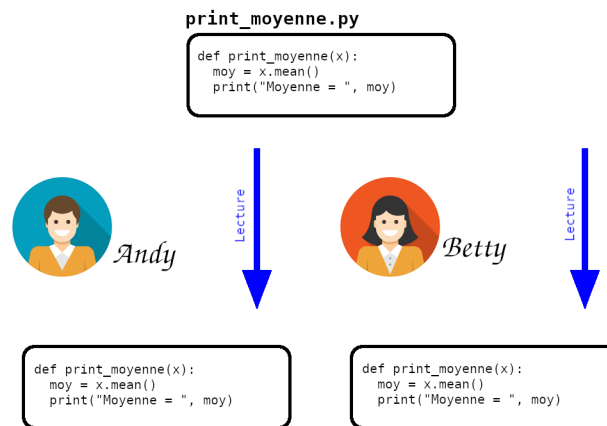
Lecture et écriture via le système de gestion de version.

## Principe général - Problème



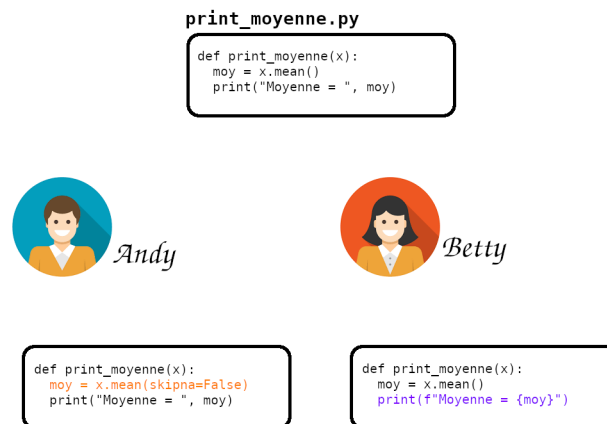
Andy et Betty veulent accéder au **même fichier** du dépôt.

## Principe général - Problème



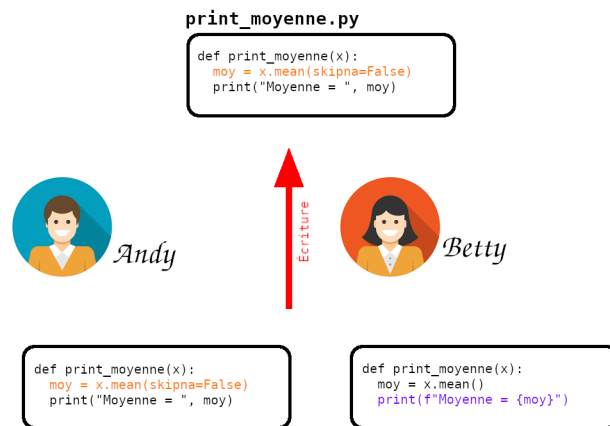
Andy et Betty **copient** le fichier chez eux.

## Principe général - Problème



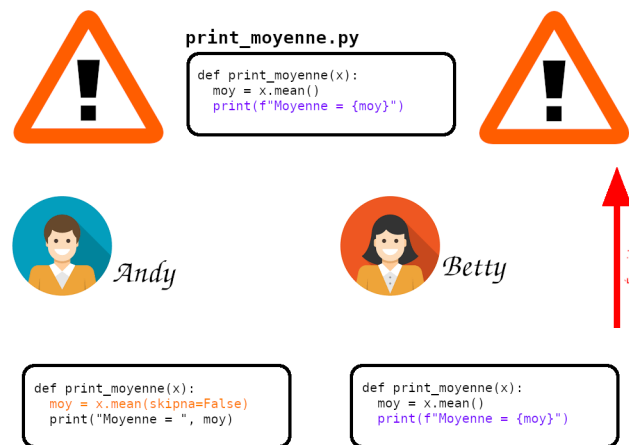
Andy et Betty font **chacun des modifications**.

## Principe général - Problème



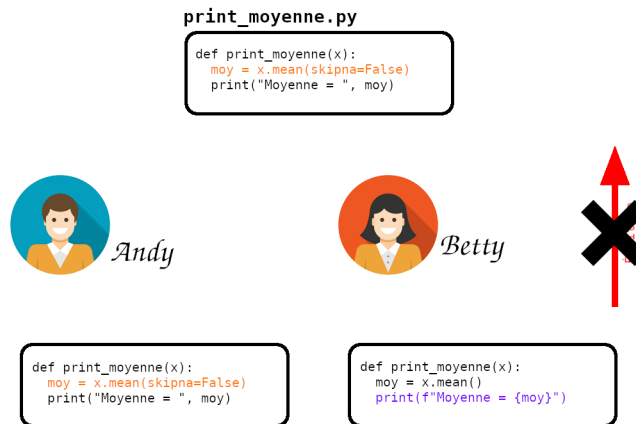
Andy écrit sur le dépôt.

## Principe général - Problème



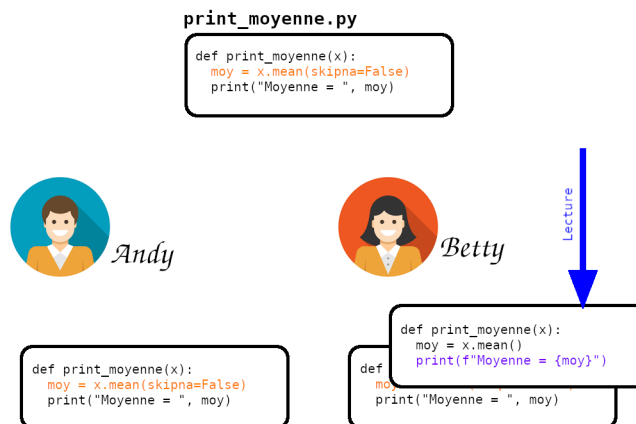
Si Betty écrit sur le dépôt, elle écrase la version de Andy !

## Principe général - Solution



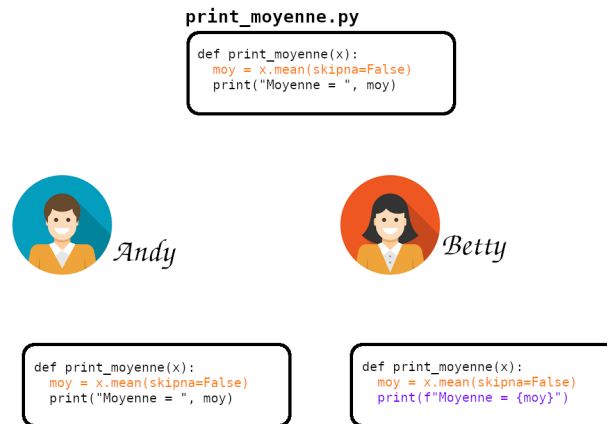
Betty ne peut pas écrire sur le dépôt car elle n'est **pas à jour**.

## Principe général - Solution



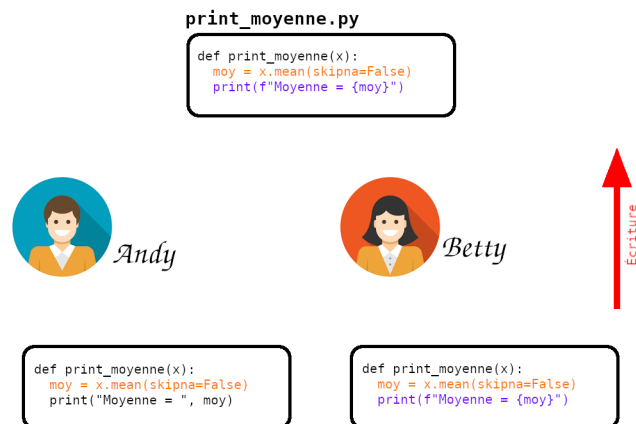
Betty se met à jour **sans perdre ses modifications**.

## Principe général - Solution



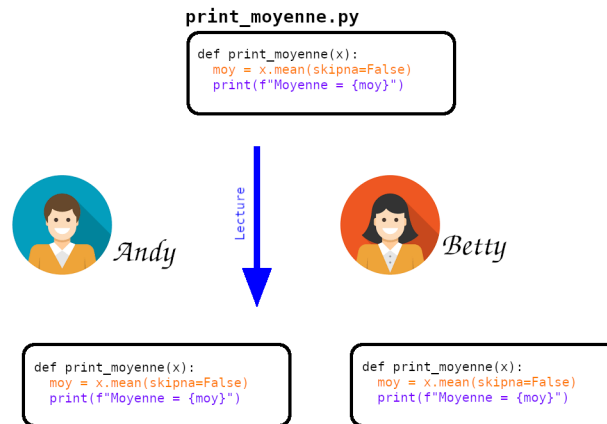
Betty **fusionne** la version du dépôt avec la sienne.

## Principe général - Solution



Betty peut **écrire** cette nouvelle version sur le dépôt.

## Principe général - Solution



Andy récupère la nouvelle version.

## Principe général - Bilan

Un système de gestion de version gère un mécanisme de **lecture-fusion-écriture** :

- Les accès (lecture et écriture) se font via le système de gestion de version.
- Le système de gestion de version conserve l'historique.
- La fusion automatique n'est possible que si
  - elle concerne un fichier texte (utilisation de **diff**),
  - les modifications ne touchent pas à la même chose.

## Git

Git est un logiciel de gestion de version (décentralisé) créé en 2005 par Linus Torvalds.

C'est un **logiciel libre** distribué sous licence GNU GPLv2.

- Git est multi-système (Linux, Mac, Windows, ...)
- Git est très utilisé (documentation, forum, ...)
- Git est sécurisé (protocoles HTTPS, SSH, ...)



# git

## Git - Vocabulaire

`add` Ajoute un élément à la *zone d'index* (*staging area*), cet élément est dit **staged**.

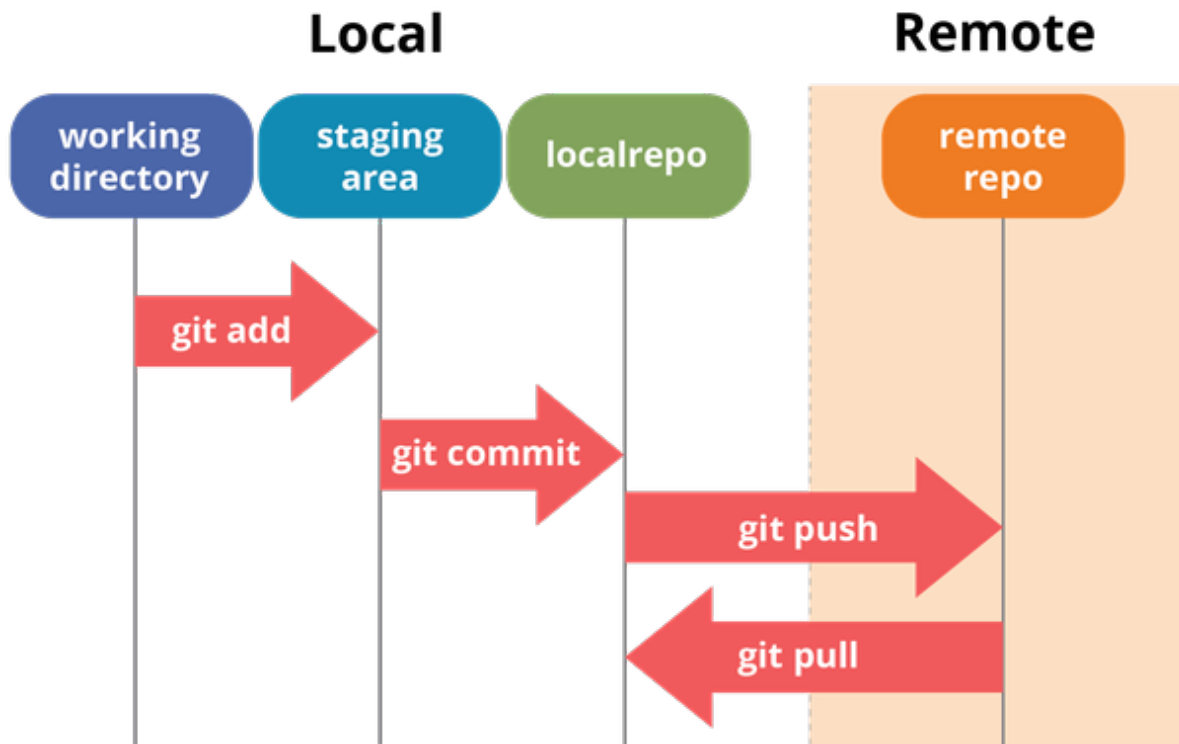
`commit` Valide les modifications dans la zone d'index.

`push` Pousse les modifications vers le dépôt distant.

`pull` Tire les modifications du dépôt distant vers le dépôt local.

Un fichier peut être **modifié**, **indexé** et **validé**.





## Git - Fichiers ignorés

Par défaut, un nouveau fichier ne fait pas partie de la zone d'index. Son état est dit **Untracked** et un tel fichier **n'est jamais poussé** sur le dépôt distant.

Le système de suivi de Git (**status**) signale ces fichiers.

De tels fichiers sont parfois nécessaires (fichiers temporaires, fichiers liés à la session en cours, ...). Pour signifier à Git qu'ils peuvent être ignorés, ils doivent apparaître dans un fichier spécial du dépôt appelé **.gitignore**.

```
__pycache__/  
*.py[cod]
```

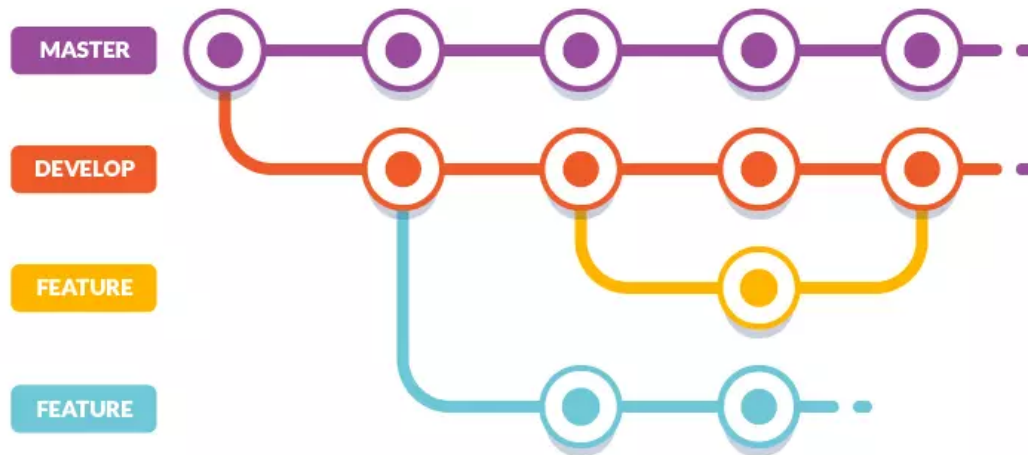
## Git - Les branches

Une **branche** est une version du projet qui suit son propre développement (avec son historique dans le système de gestion de version) et qui peut être **fusionnée** avec d'autres branches.

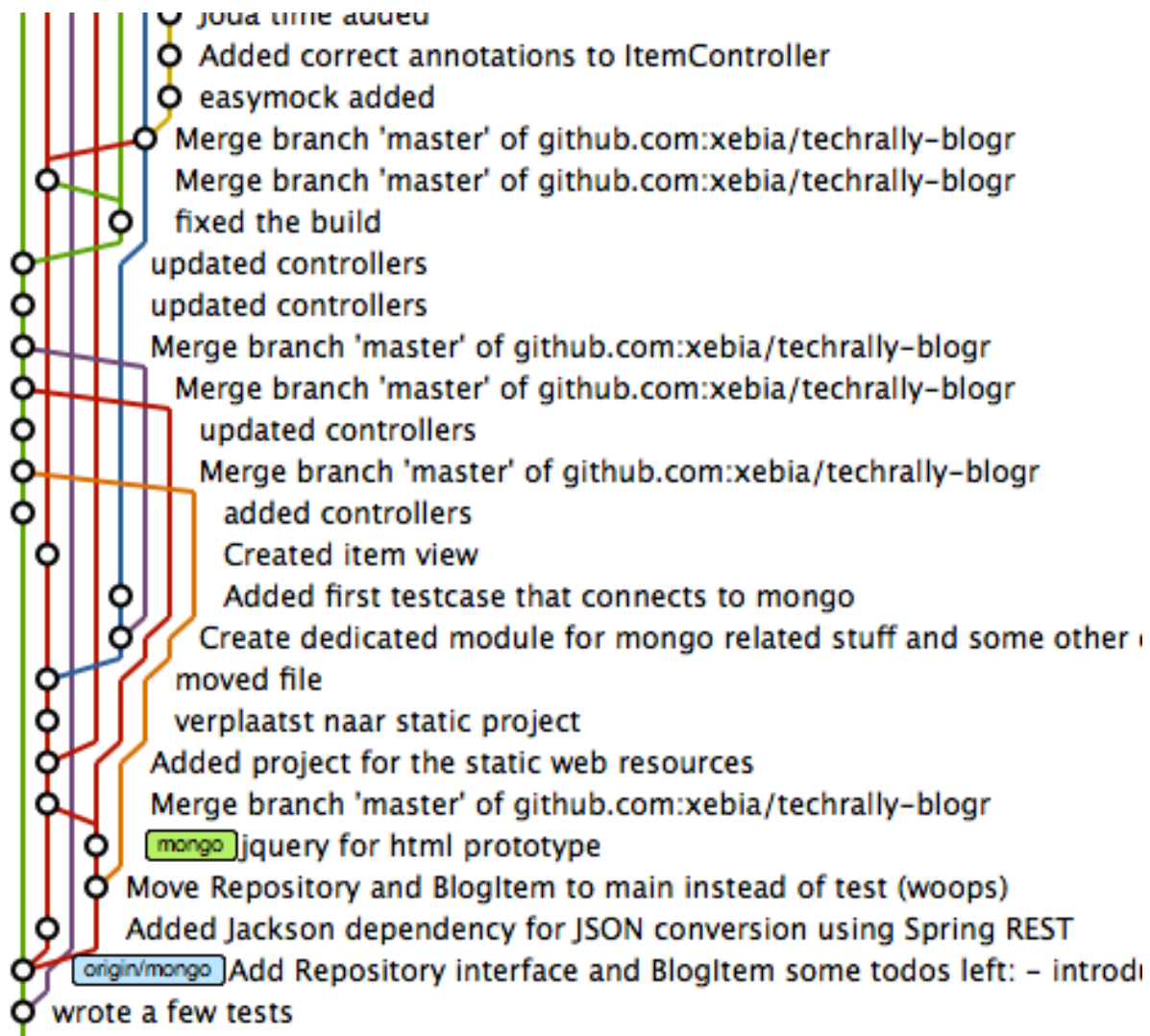
Exemples classiques :

- Branche **main** (ou **master**) : version stable du projet qui peut être utilisée en production.
- Branche **dev** : version de développement, instable par définition mais plus avancée que la version stable du projet.
- Branche **bug42** : version dédiée à la correction d'un bug.

## Git - Les branches (Cas simple)



## Git - Les branches (La réalité...)



## Git - Vocabulaire des branches

`branch test` Crée une nouvelle branche `test` dans le dépôt local.

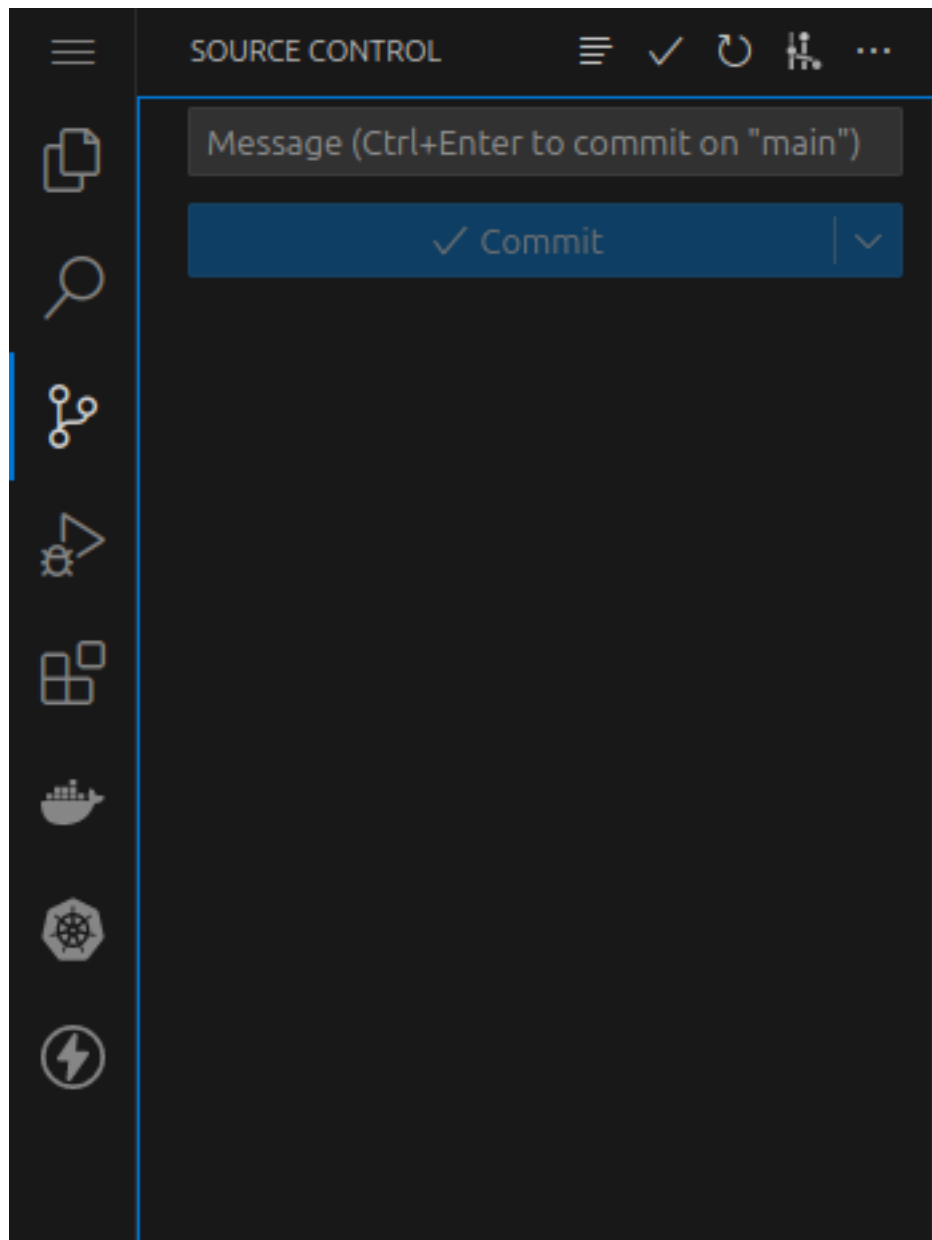
`checkout test` Bascule vers la branche `test` du dépôt local.

`merge test` Fusionne la branche `test` du dépôt local avec la branche courante.

Une fusion peut être bloquée en cas de conflit entre les modifications. Git offre des outils pour gérer cela mais ce travail reste essentiellement manuel.

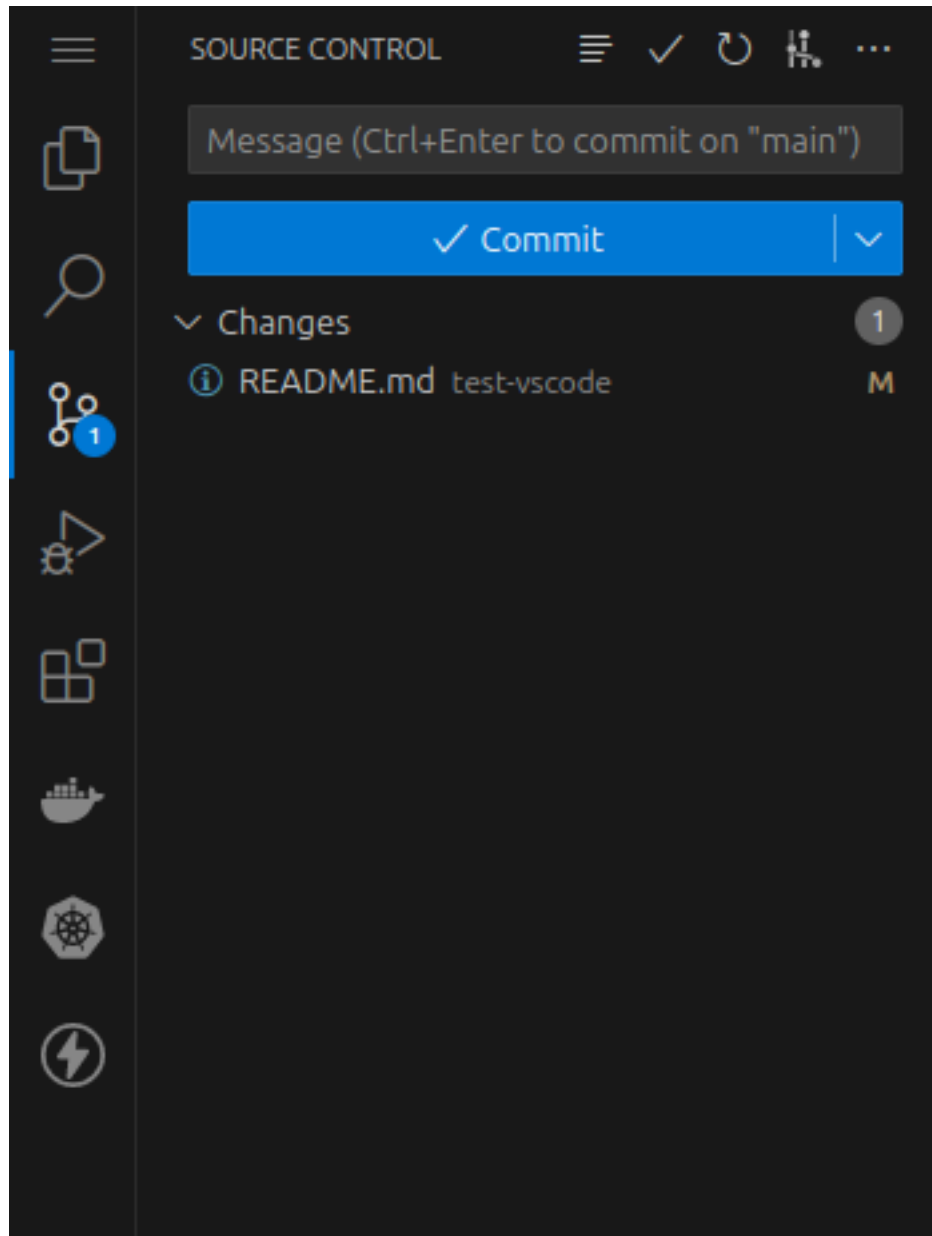
(Il est bien entendu possible de pousser une branche sur le dépôt distant.)

## Git et VSCode



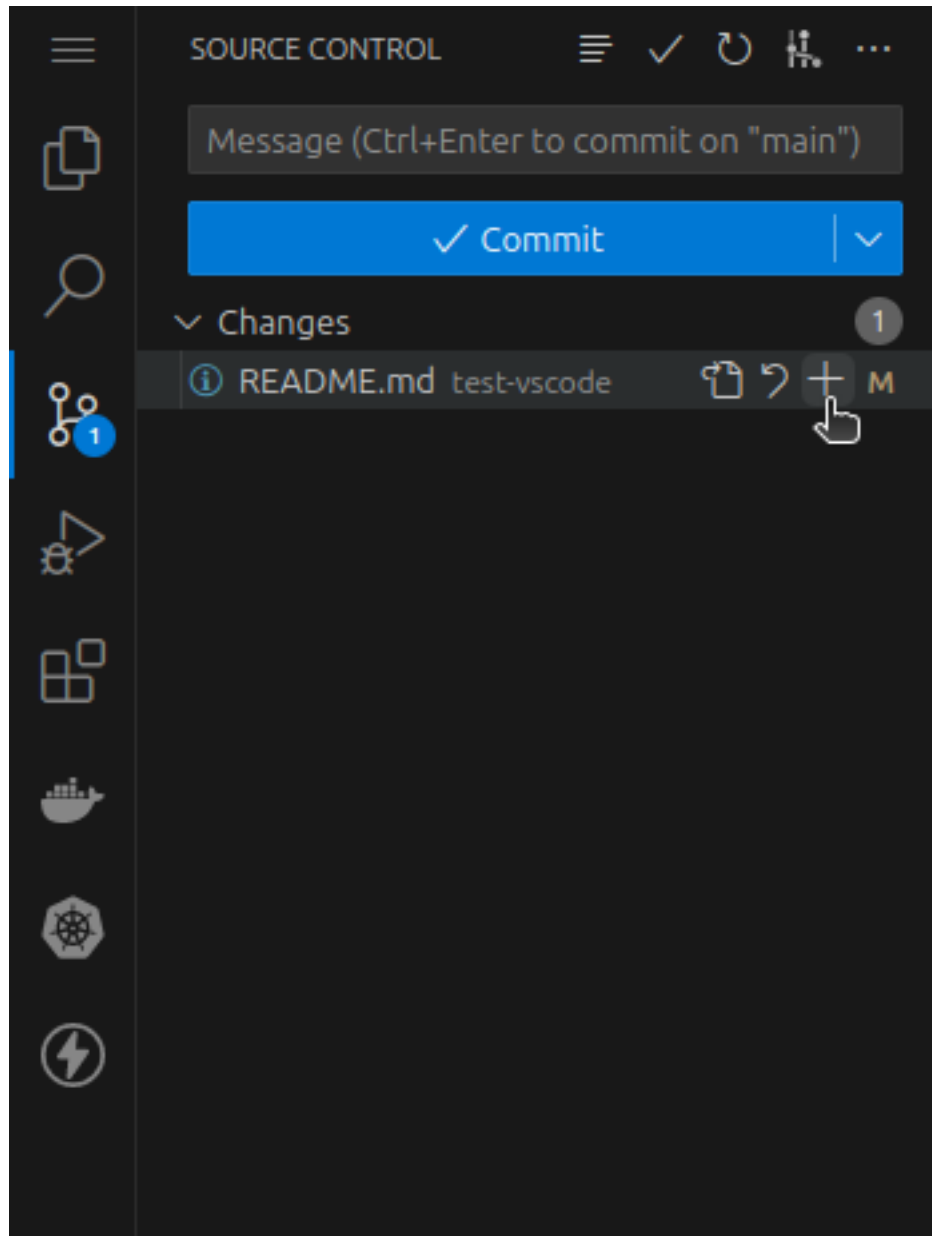
Quand le projet est suivi par Git, un menu *Source Control* est disponible dans la barre de gauche.

## Git et VSCode



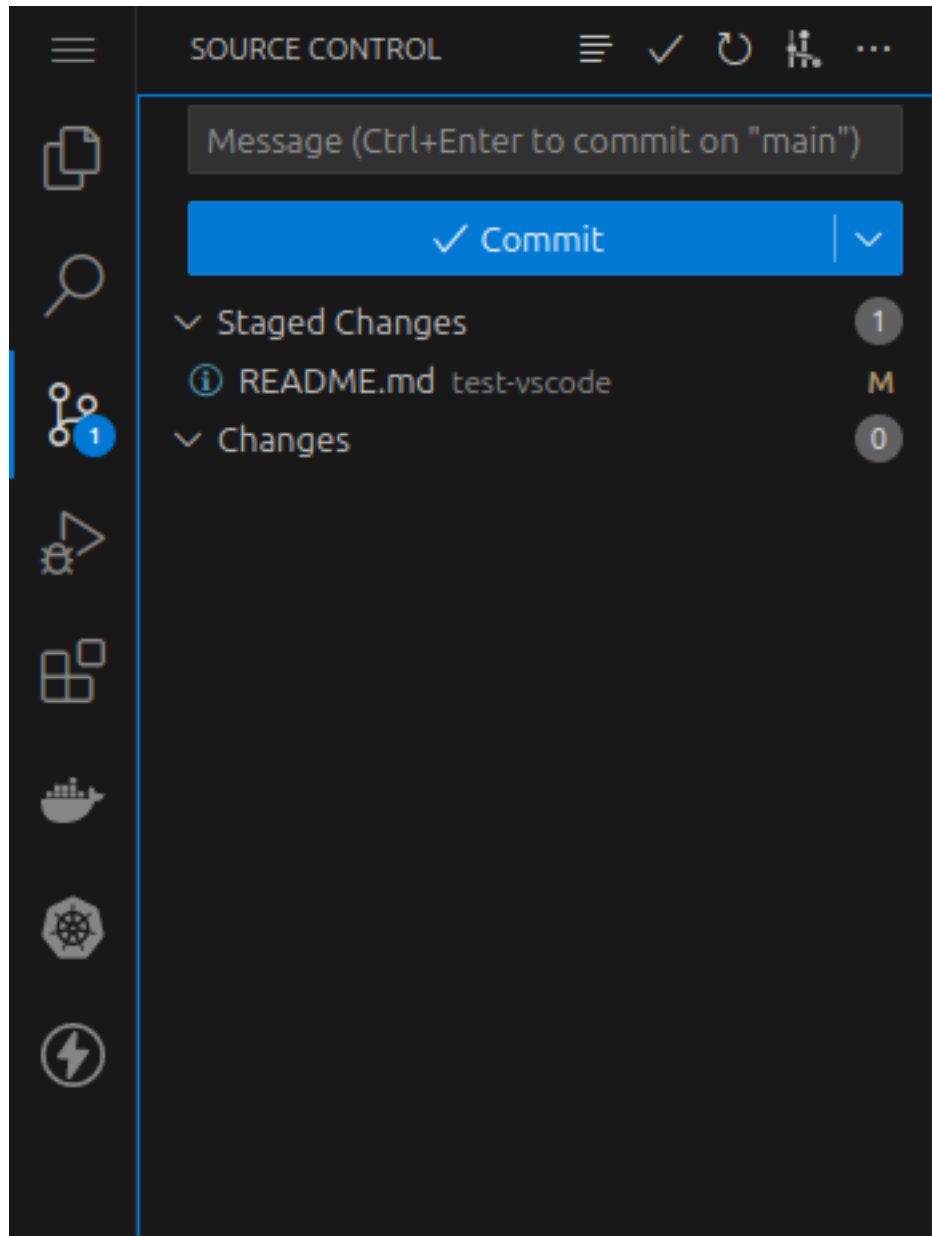
Lorsque un fichier est modifié (M), supprimé (D), renommé (R), non suivi (U), ..., il apparaît dans la liste.

## Git et VSCode



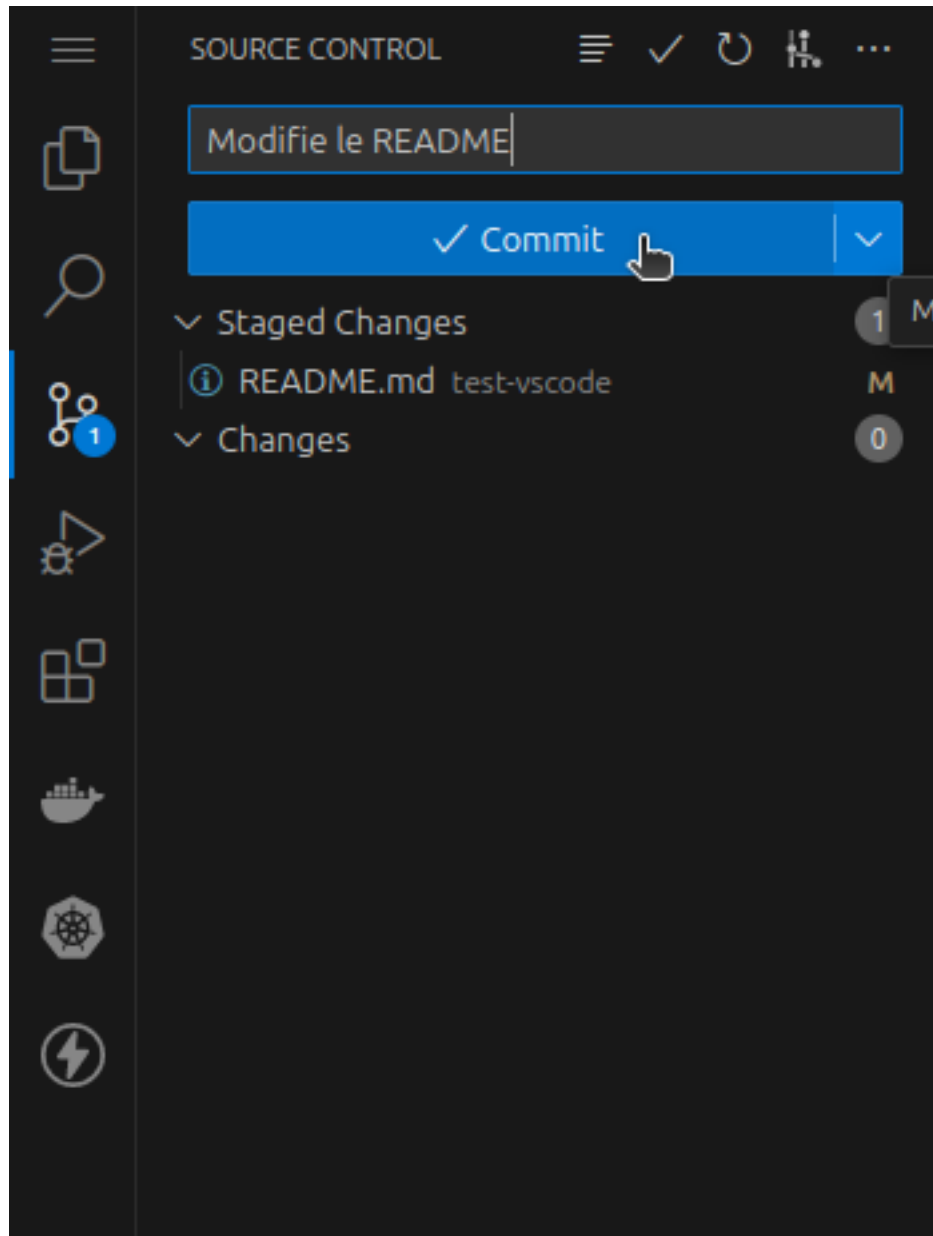
Un ou plusieurs changements peuvent être ajoutés à la *zone d'index* grâce au bouton + associé.

## Git et VSCode



Les changements sont maintenant *staged* et ils peuvent constituer un *commit*, *i.e.* une étape de notre projet.

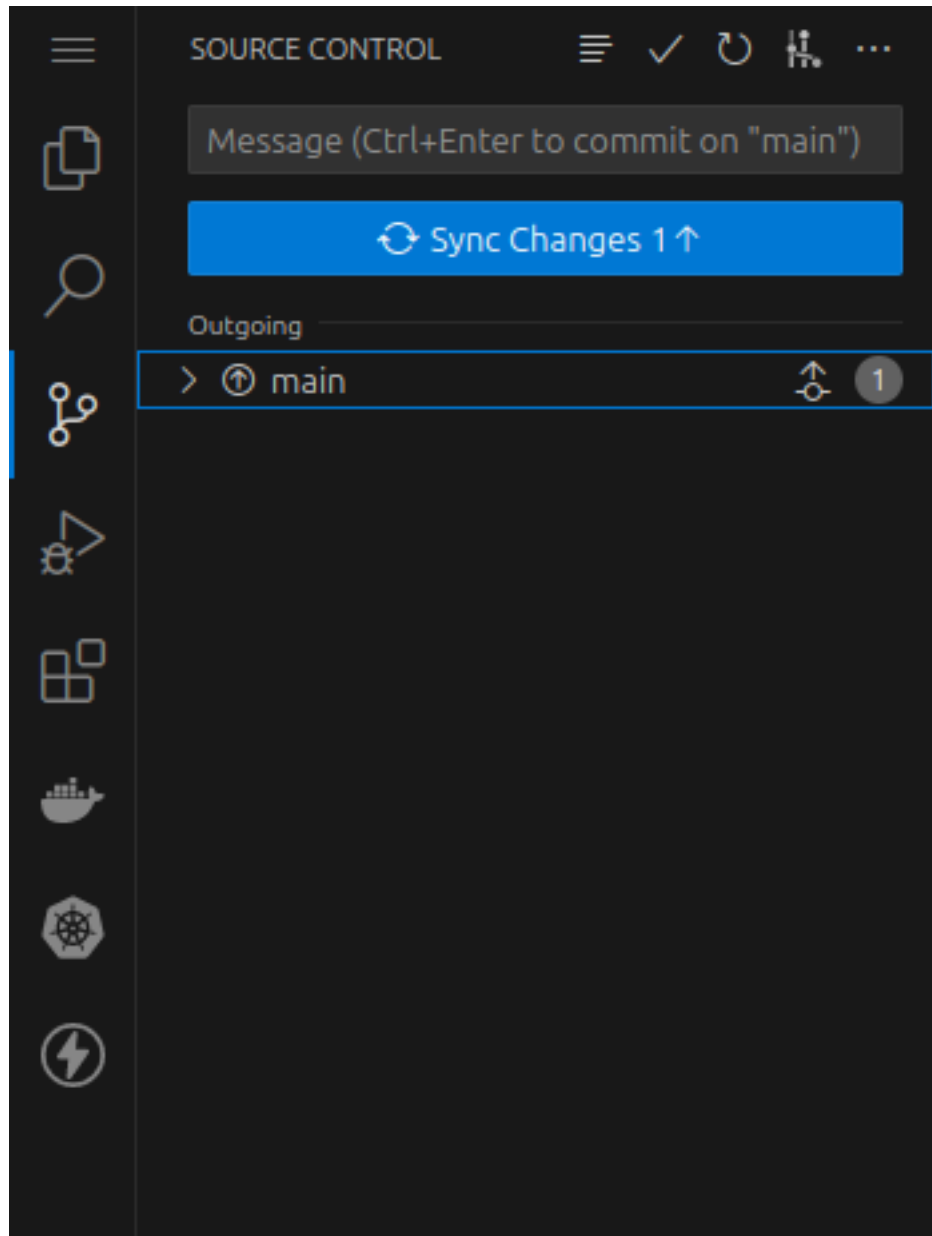
## Git et VSCode



Avec une brève description pour faciliter la maintenance du projet, le commit est créé avec le bouton Commit.

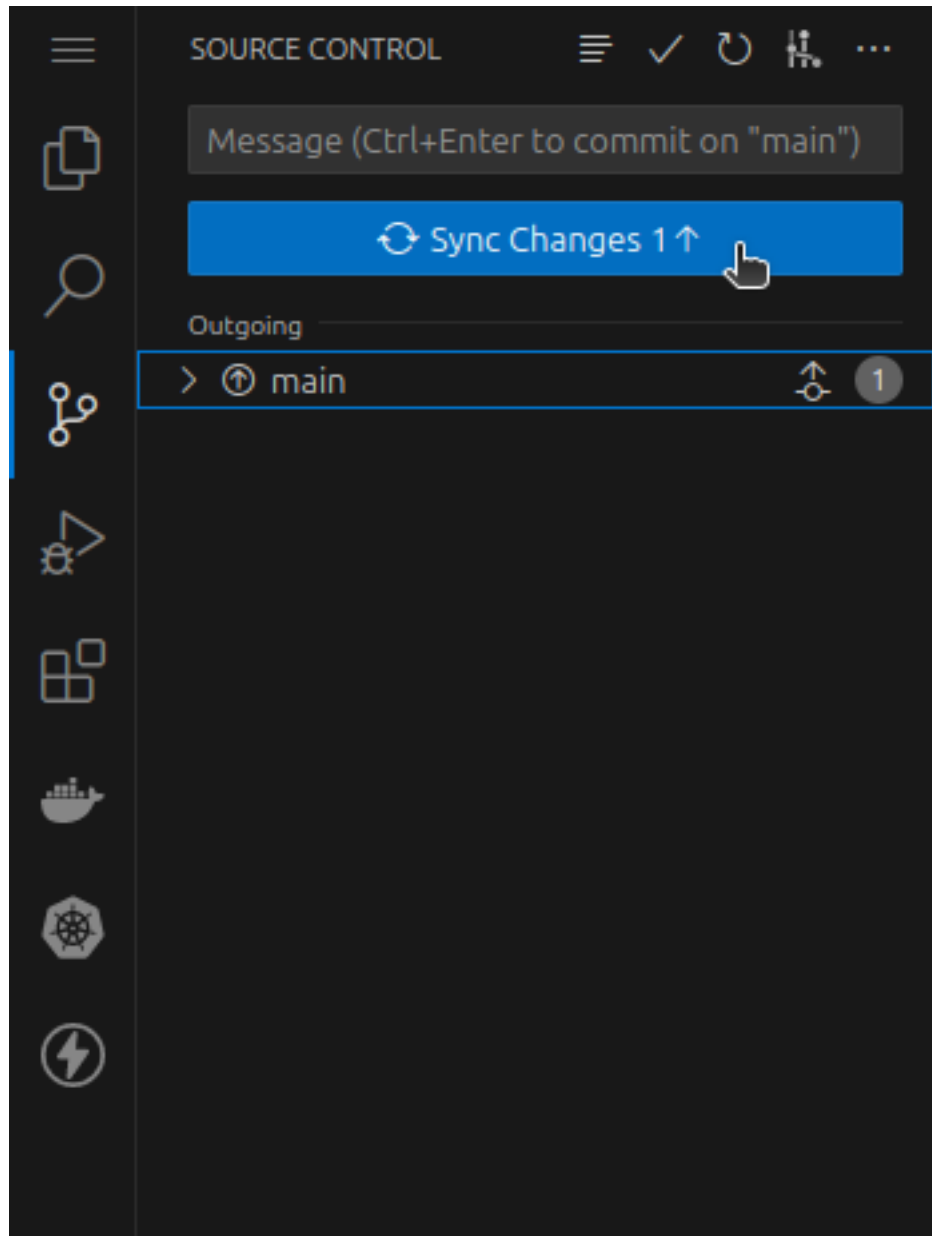


## Git et VSCode



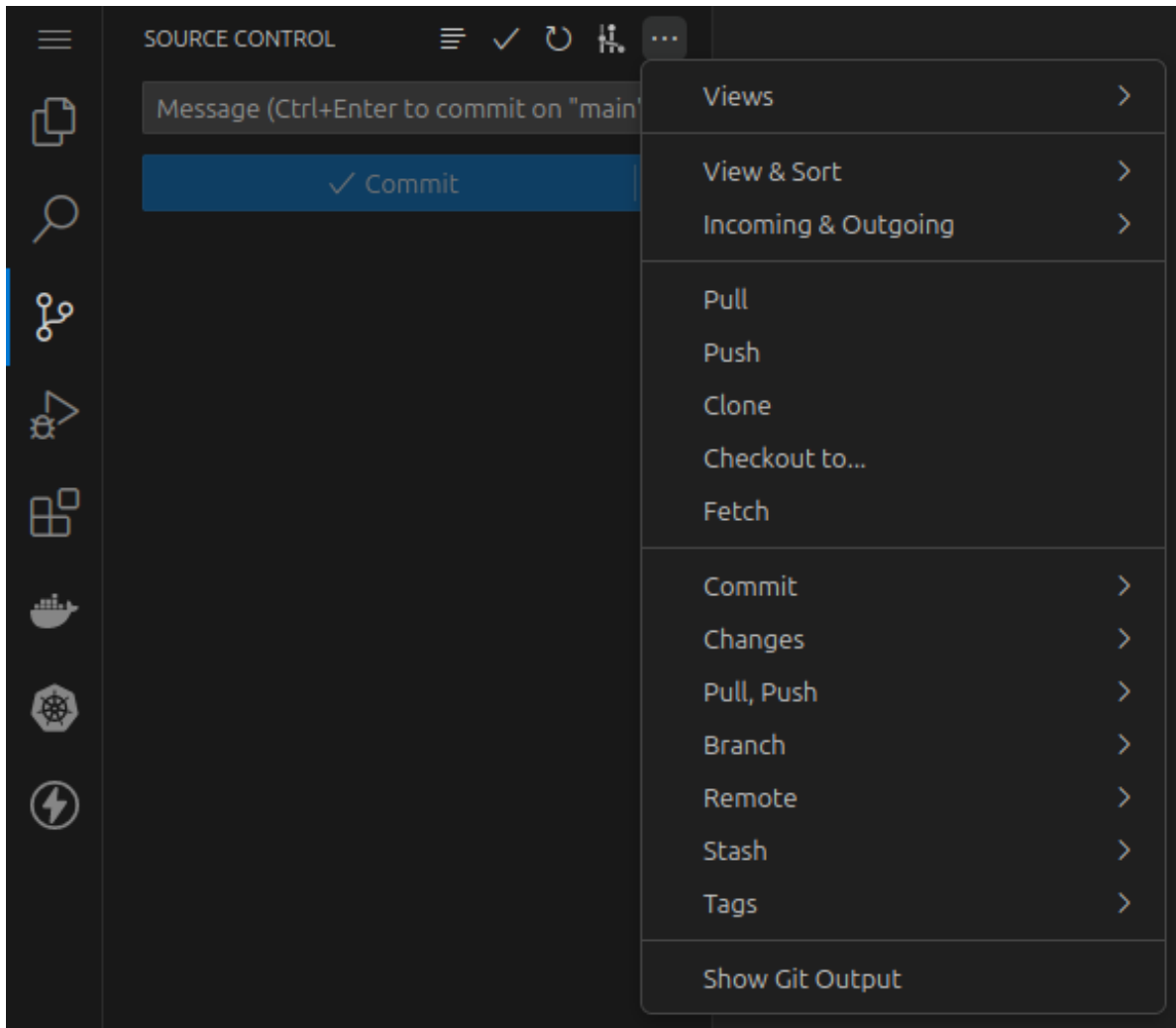
Un commit a été ajouté à la branche `main` dans cet exemple dont le détail apparaît dans la zone inférieure.

## Git et VSCode



Si le projet est stocké dans un *dépôt distant*, les changements peuvent être poussés (**push**) avec **Sync Changes**.

## Git et VSCode



Au-delà de ces opérations courantes, le menu ... offre bien d'autres possibilités (pull, branch, ...).

### Git et VSCode - Ligne de commande

Si les outils mis à disposition ne suffisent pas, il est possible d'interagir via la *ligne de commande* dans un terminal accessible via le raccourci **Ctrl+J**.

Ces manipulations sont destinées à un **usage plus avancé**.

Par exemple, pour faire un **commit**, Git a besoin de savoir qui nous sommes :

```
git config --global user.name "Votre Nom"
git config --global user.email "votre@mail.net"
```

Ces informations seront particulièrement utiles pour les projets **hébergés à distance** (voir la configuration Git de VSCode avec Onyxia dans un instant).

## Moralité

L'utilisation d'un système de gestion de version est :

- **indispensable** pour travailler à plusieurs,
- **sécurisant** même lorsqu'on est seul sur le projet,
- **simple** car l'effort à fournir est négligeable.

**Ne pas utiliser un système de gestion de version est une faute professionnelle.**

Pour aller plus loin : `tag`, `revert`, `rebase`, ...

## GitHub

GitHub est un **service web d'hébergement** et de **gestion** de dépôts distants basés sur le logiciel Git.

GitHub propose des comptes gratuits et une offre payantes pour des usages plus avancés.



Un dépôt peut être **public** ou **privé**, GitHub permet le contrôle des accès par les utilisateurs.

GitHub offre aussi de nombreux services : suivi de bugs, forum, wiki, pages web, ...

GitHub a été racheté par Microsoft en 2018 pour 7,5 milliards \$.

### **Profil GitHub**

Une fois connecté, vous découvrez l'interface.

The screenshot shows a GitHub profile page for a user named 'FormateurCepe'. The profile picture is a circular avatar with a pink and white pixelated design. The username 'FormateurCepe' is displayed below the avatar, along with an 'Edit profile' button and a note that the user joined 3 minutes ago. The main content area features a 'Popular repositories' section with the message 'You don't have any public repositories yet.' Below this is a '1 contribution in the last year' section, which includes a contribution graph for the last year (April to December). The graph shows a single green square in December, indicating one contribution. A legend below the graph explains that the color of the squares represents the number of contributions, with a scale from 'Less' to 'More'. A text box below the graph provides information about the contribution graph and includes a link to 'Read the Hello World guide'.

Search or jump to... Pulls Issues Codespaces Marketplace Explore

Overview Repositories Projects Packages Stars

Popular repositories Customize your pins

You don't have any public repositories yet.

FormateurCepe Edit profile

Joined 3 minutes ago

1 contribution in the last year Contribution settings

Apr May Jun Jul Aug Sep Oct Nov Dec

Learn how we count contributions Less More

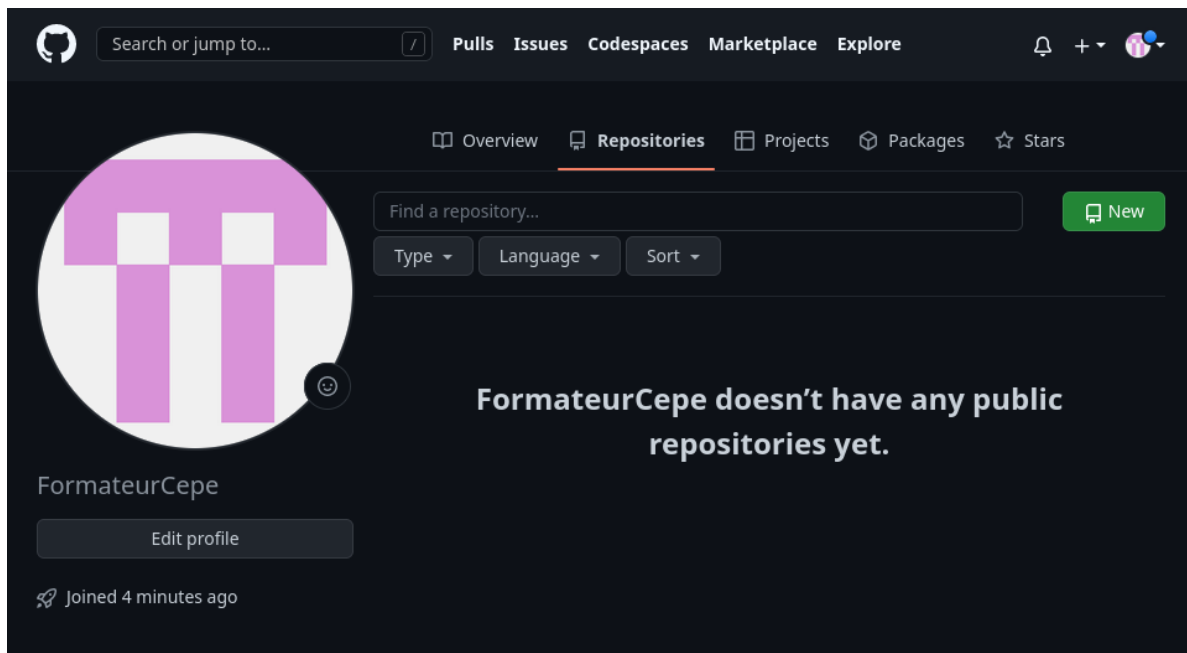
This is your **contribution graph**. Your first square is for joining GitHub and you'll earn more as you make [additional contributions](#). More contributions means a darker green square for that day. Over time, your chart might start looking [something like this](#).

We have a quick guide that will show you how to create your first repository and earn more green squares!

[Read the Hello World guide](#)

## Héberger ses dépôts sur GitHub

Vous pouvez voir vos dépôts ou en créer de nouveaux.



## Créer un dépôt sur GitHub

Interface de création pour :

- donner un **nom** à votre dépôt,
- le déclarer **public** ou non,
- créer certains **fichiers utiles** : le fichier README **initialise** le dépôt,
- choisir une **licence** (libre, naturellement).

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner \*



FormateurCepe ▾

Repository name \*

/

Great repository names are short and memorable. Need inspiration? How about **congenial-palm-tree?**

Description (optional)



**Public**

Anyone on the internet can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.

### Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more](#).

### Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: None ▾

### Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

License: None ▾

You are creating a public repository in your personal account.


Create repository

## Paramètres de son compte GitHub

Vous pouvez configurer de nombreux paramètres dans l'interface :





- vos informations personnelles (l'**adresse mail** est la plus importante car elle apparaît dans chaque commit),
- l'apparence de l'interface (il faudra choisir son côté de la Force),
- votre **clé SSH** (obligatoire pour pouvoir pousser des modifications en l'absence de *token*),
- ...





FormateurCepe


Your personal account

 Public profile


 Account


 Appearance


 Accessibility


 Notifications


Access


 Billing and plans


 Emails

 Password and authentication


 Sessions


 SSH and GPG keys


 Organizations


 Moderation


Code, planning, and automation


 Repositories

 Codespaces


 Packages

 Copilot


 Pages


 Saved replies

Security


 Code security and analysis


Integrations


 Applications

 Scheduled reminders

Archives

 Security log

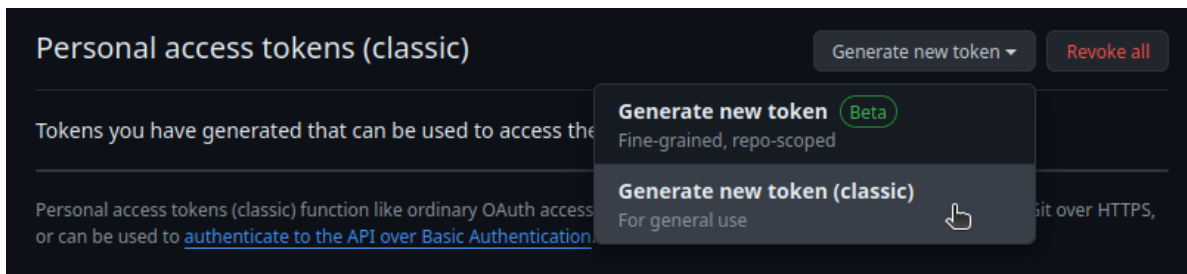
 Sponsorship log

 Developer settings

## Obtenir un token pour GitHub

Un *token* (ou jeton d'accès) est un mécanisme d'identification sécurisé pour interagir de manière fluide avec vos dépôts. Une alternative est l'utilisation d'une clé SSH.

La [page de gestion des tokens](#) permet de générer un nouveau token. Nous utiliserons le format *classique*.



## Obtenir un token pour GitHub

### New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

**Note**

Onyxia

What's this token for?

**Expiration** \*

30 days

 The token will expire on Fri, Jul 19 2024

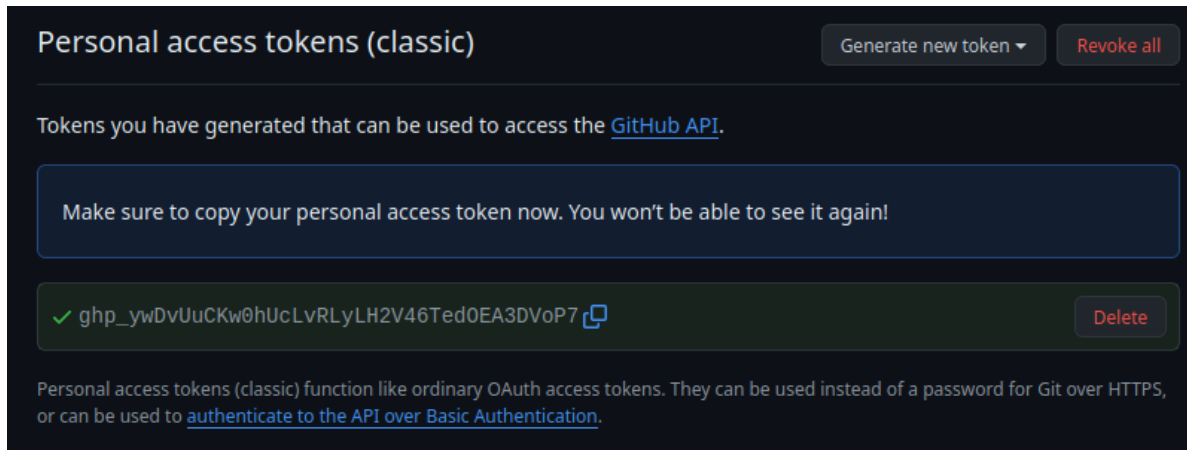
**Select scopes**

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry

- Donner un nom à votre token.
- Définir une durée de validité (par sécurité).
- Limiter la portée à **repo** uniquement pour éviter les problèmes en cas de fuite.

## Obtenir un token pour GitHub

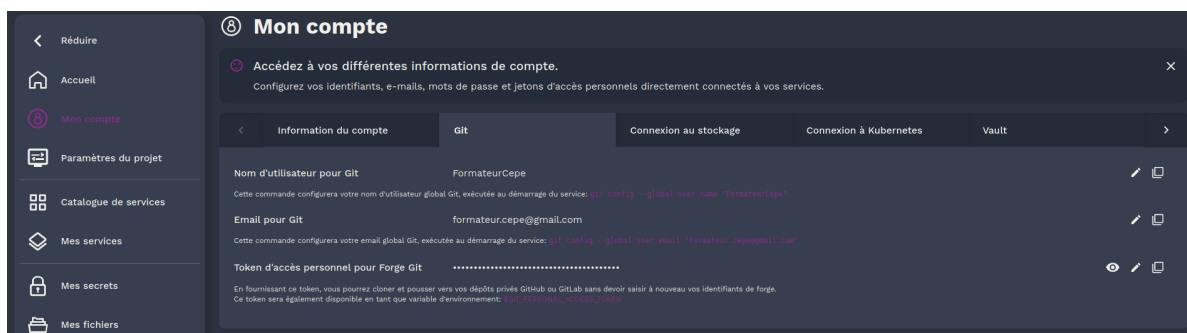


Le token est créé ! Il doit être copié et stocké en lieu sûr (par exemple, sur Onyxia) car **il ne pourra plus être affiché**.

## Configuration Git avec Onyxia

Par défaut, un compte sur la forge **GitGenes** a été créé mais nous utiliserons *GitHub* dans la suite (le même principe reste valable pour d'autres forges basées sur Git).

Une configuration **globale** du nom, du mail et du token est possible depuis l'onglet *Git* du menu *Mon compte* de Onyxia.




Une configuration **locale** avec l'adresse du dépôt (et la branche en option) est disponible à la création du service.

Cr  er votre propre service

R  initialiser les configurations

Copier l'URL de lancement automatique

Enregistrer la configuration

 Vscode-python

Nom personnalis  

vscode-python

Version

1.11.35

Annuler

Lancer

Configuration Vscode-python

<

Git

Discovery

Service

Persistence

>

Git user configuration

☒ Enabled

Add git config inside your environment

Name

FormateurCepe

Email


formateur.cepe@gmail.com

Cache

0

Token

.....



Repository

https://github.com/FormateurCepe/mon\_depot.git

Branch

Branch automatically checked out

   vous de jouer !