



Introduction à Git et Github

Contrôler le cycle de vie de vos codes sources et projets avec git et github

CONCEPTS DU VERSIONING

Qu'est ce qu'un système de gestion de version ?

Un gestionnaire de version (**VCS** en anglais pour **V**ersion **C**ontrol **S**ystem) est un outil qui permet de **tracker l'historique** des modifications d'un groupe de fichiers au cours du temps.

Il permet de se **replacer** sur n'importe quelle version à tout moment. Les utilisateurs peuvent consulter l'historique pour découvrir :

- Ce qui a été modifié dans le projet.
- Par qui ?
- Quand ?
- Pourquoi ?



Principaux CVS



Développé au milieu des années 80, il est le précurseur des VCS. CVS est un **CVCS** fonctionnant sur des systèmes de type **Unix**. CVS est **gratuit** et **open source**.



Mercurial a été développé en même temps que Git pour concurrencer ce dernier comme CVS pour le développement du noyau Linux.

Il a eu moins de succès que Git, mais est utilisé sur des projets majeurs comme **OpenOffice**.



SVN a été créé en 2000 pour servir d'alternative à CVS. Une des évolutions majeures fût l'introduction du concept d'opérations **atomiques**.



Git a été créé en 2005 par **Linus Torvalds**, créateur du noyau du système d'exploitation **Linux**.

Il a développé git pour faciliter l'intégration des contributions au projet Linux.

Principaux modèles d'architectures distantes de VCS.

CVCS

Avantages :

- Partage des versions automatisés.
- Gestion des autorisations plus simple.
- Optimisation de l'espace de stockage.

Inconvénients :

- Point de défaillance centrale et perte de données.
- Dépendance à la connexion réseau.

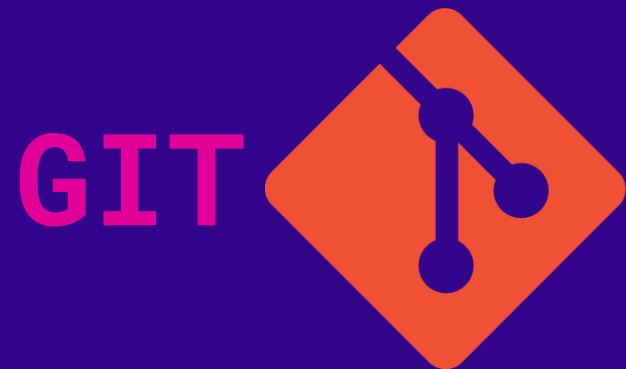
DVCS

Avantages :

- Travail hors connexion
- Architecture réseau à tolérance de panne.
- Facilite la collaboration.

Inconvénients :

- La copie initiale du projet est plus lente.
- Nécessite plus d'espace de stockage pour chaque client.
- Prise en charge insuffisante des fichiers binaires.



Git

Git est le **SGV** distribué le plus utilisé au monde par les développeurs¹.

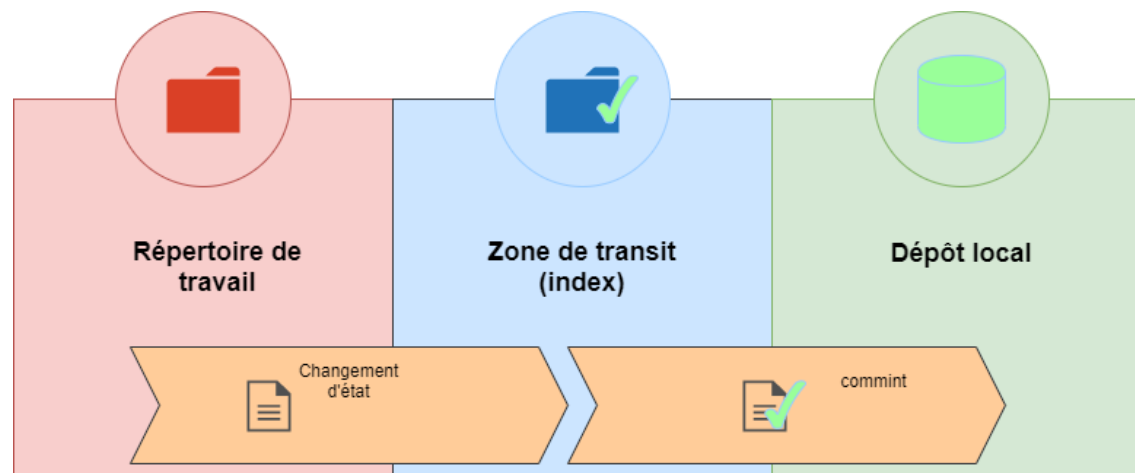
Il offre un accès à un historique complet de toutes les modifications dans un projet.

Lorsqu'il est intégré à un projet, Git gère l'organisation du stockage des fichiers versionnés.

Les actions de Git ont un impact direct sur l'état du répertoire courant sur la machine.

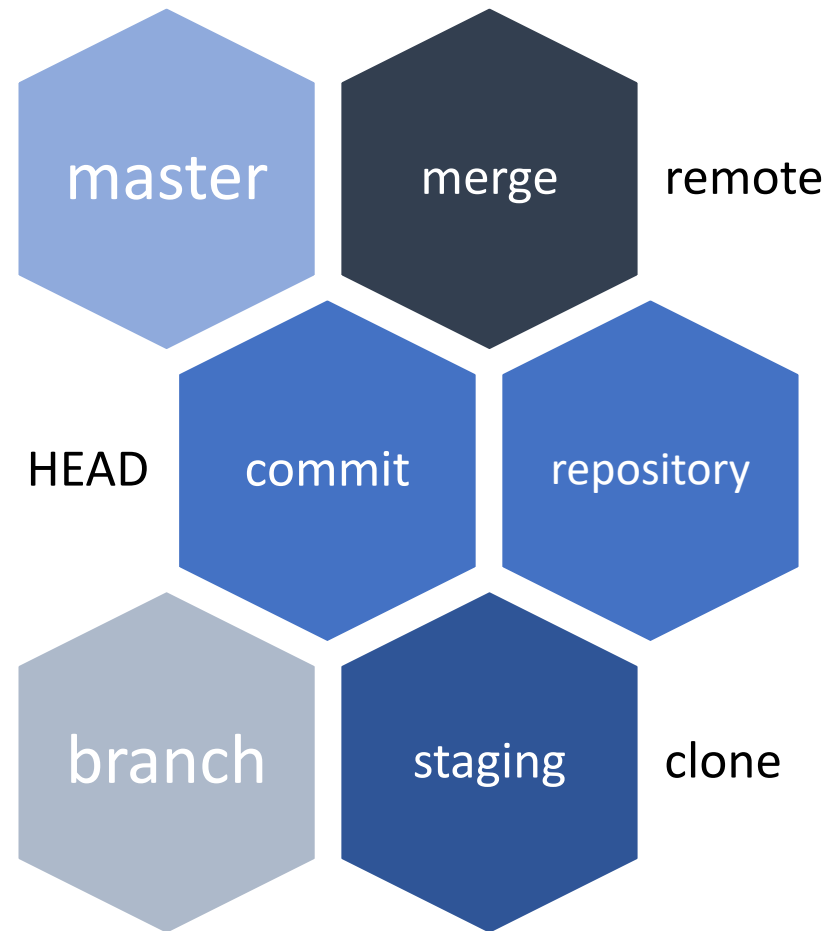


Architecture de git



- **Un dépôt git comprend trois principaux segments :**
 1. **Répertoire de travail (working directory) :** ensemble de fichiers et dossiers correspondant à la version courante.
 2. **Zone de transit (index ou staging en anglais) :** L'ensemble des modifications en attentes d'être intégrées au dépôt lors la prochaine validation (commit).
 3. **Le dépôt (Repository en anglais) :** ensemble des éléments du projet, historiques des modifications, les validations de changements (commit).

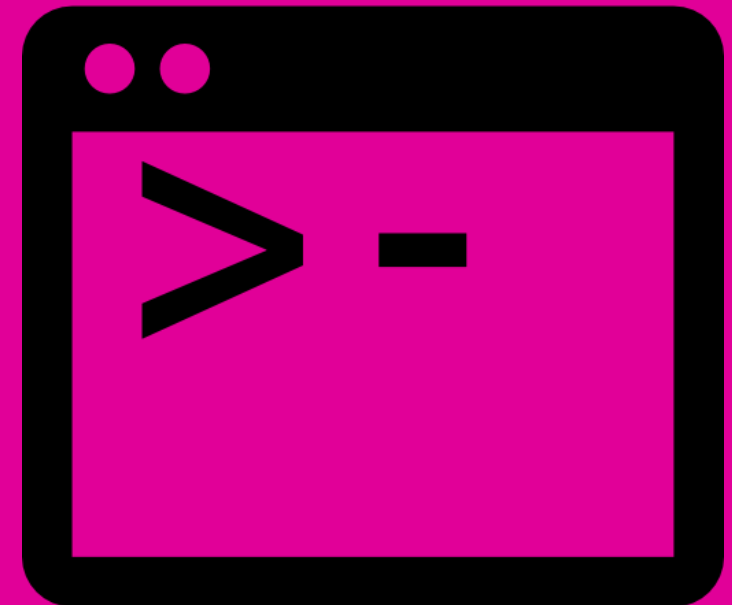
Terminologie git



FOCUS SUR LES COMMANDES DE BASE

Syntaxe

- L'interface en ligne de commande (cmd) est un outil essentiel pour exploiter pleinement toutes les fonctionnalités proposées par Git.
- Bien qu'il existe des outils graphiques pour Git, ils sont généralement limités en termes de fonctionnalités.
- **Syntaxe commande git**
- # Format strict
- `git <command> --<option>=<param>`
- #Format light
- `git <command> --<option> <param>`



Configuration de git

git config

Elle sert à configurer divers paramètres de Git, tels que le nom d'utilisateur, l'adresse e-mail, etc....

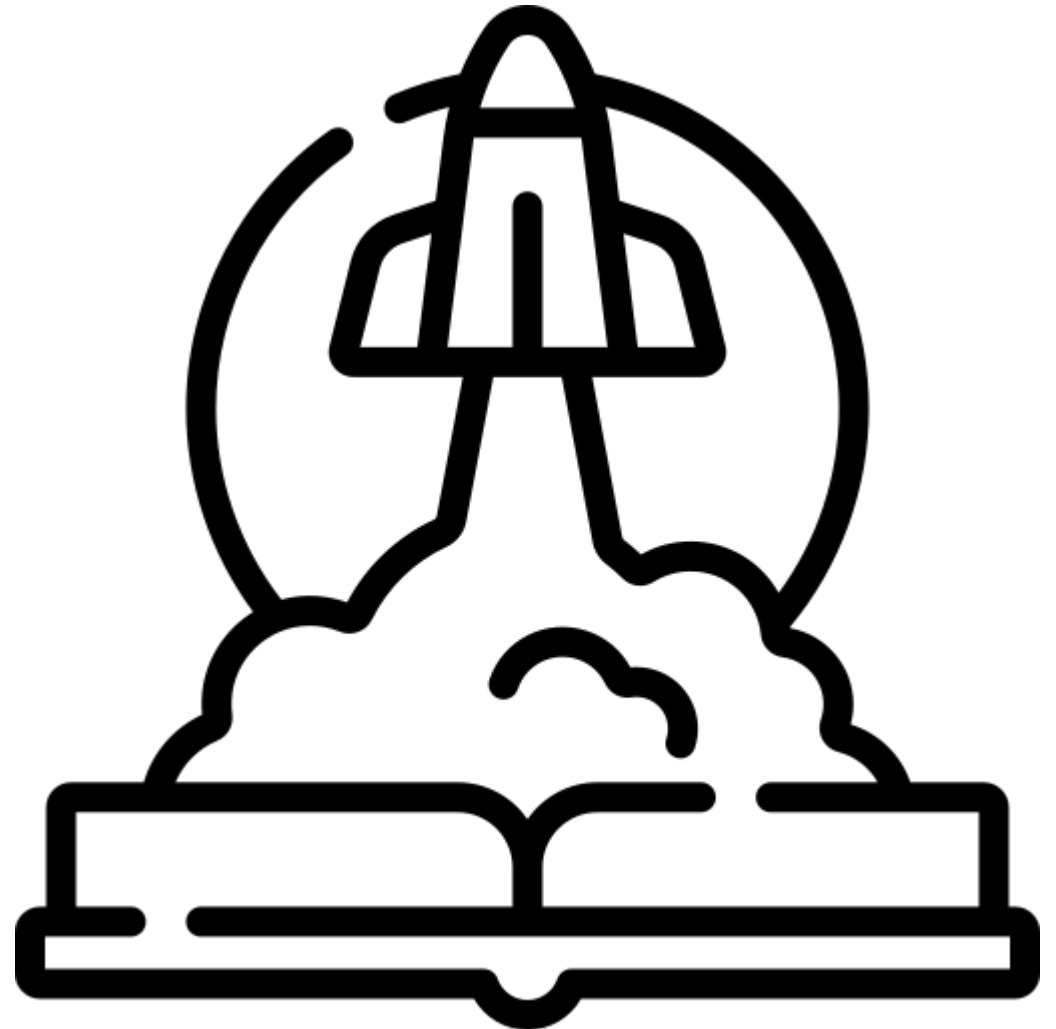
Elle permet de personnaliser le comportement de Git sur votre système.

Un utilisateur ne peut pas réaliser de commit sans avoir configuré ses informations d'identification au préalable.

Configuration de l'identité

```
# configuration du nom
git config --global user.name "John DOE"

# configuration de l'adresse email
git config --global user.email johndoe@mail.com
```



Commandes de base git

git help

Elle permet d'utiliser l'aide git :

Afficher l'aide pour une commande

```
git <command> --help
```

```
git help <command>
```



Initialisation d'un projet git



Initialiser un dépôt git consiste à mettre en place un dépôt vide ou à convertir un répertoire existant en dépôt git.

La mise en place d'un dépôt permettra de versionner les fichiers et de suivre l'historique des modifications.

Le dépôt créé sera constitué de :

- Dossier de travail (working directory)
- Zone de transit (Staging Area) : fichier binaire
- Dépôt local (local repository)

Commande :

git init

Commandes de base git

La commande **git add** permet d'indiquer à Git les fichiers que l'on souhaite inclure dans le prochain commit.

Git se charger se charge de d'ajouter les fichiers concernés dans la zone de transit (stage) en attendant le prochain **commit**.

Ajouter des fichiers spécifiques à l'index

Demander à git de suivre des fichiers

```
git add <file1> <file2>, ...
```

Choisir les fichiers à ajouter en utilisant un motif (pattern)

Utiliser un motif pour choisir les fichiers à suivre

```
git add *.js *.css *.html
```

Ajouter tous les fichiers à l'index

Ajouter tous les fichiers du dossier courant

```
git add all / git add . / git add *
```

Commandes de base git

La commande **git commit** permet de faire une photographie de votre projet à un instant T et de l'enregistrer dans le dépôt git.

Le commit est accompagné d'un message qui décrit concisément la modification effectuée.

L'option **amend** permet de modifier le dernier commit

Permet de sauvegarder les modifications

```
git commit -m "ajout .gitignore"
```

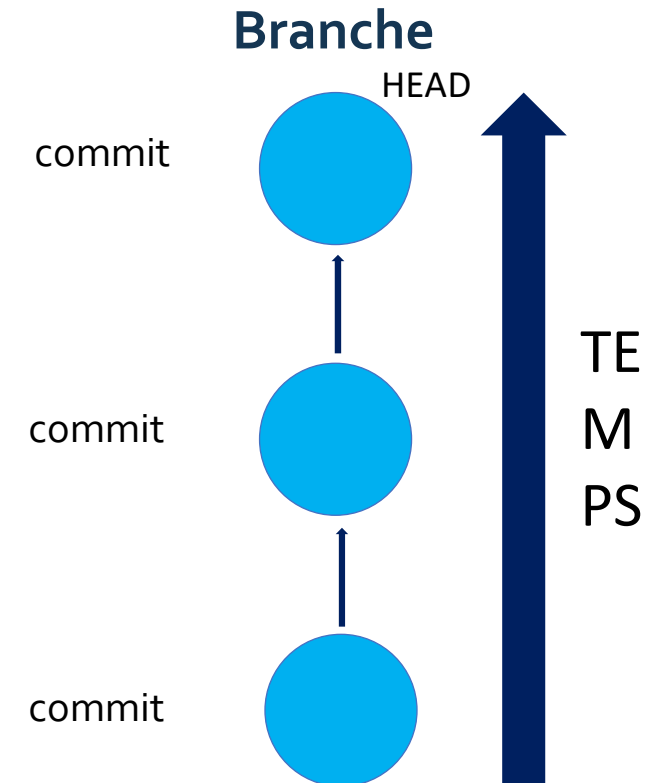
Modifier le dernier commit

Ajouter des modifications

```
git commit --amend
```

modifier le message

```
git commit --amend -m "ajout du .env au fichier .gitignore"
```



Commandes de base git

La commande **git status** permet pour déterminer l'état des fichiers du projet.

visualiser les modifications dans les fichiers

```
git status
```

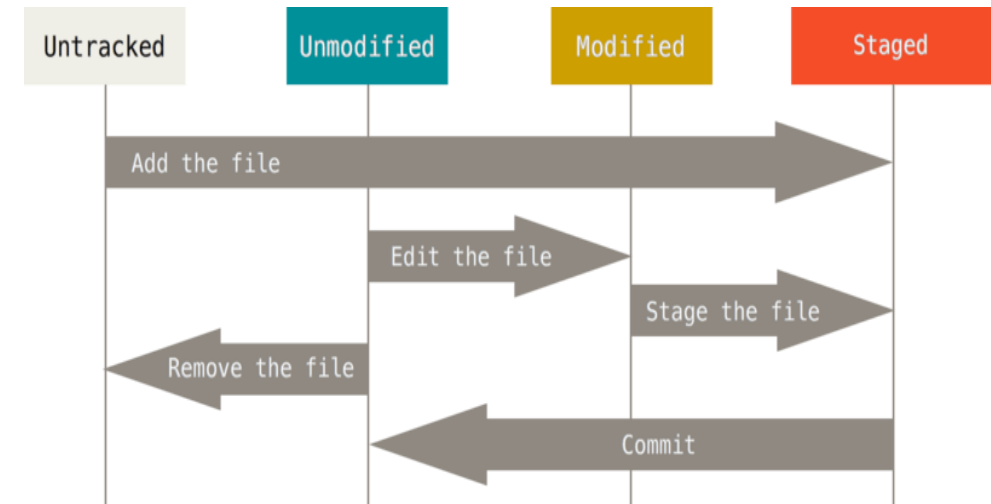
La commande **git log** permet pour déterminer l'état des fichiers du projet.

```
git log
```

La commande **git diff** permet de comparer l'état du projet entre deux commits. Elle liste l'ensemble des modifications réaliser la date de début.

visualiser les modifications dans les fichiers

```
git diff
```



<https://git-scm.com/book/fr/v2/Les-bases-de-Git-Enregistrer-des-modifications-dans-le-d%C3%A9p%C3%B4t>

Focus sur les branches git

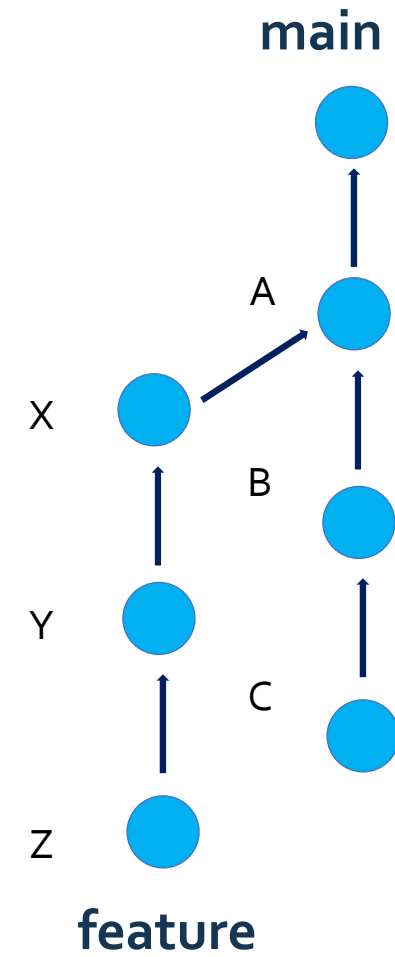
Les branches git

Les branches sont une fonctionnalité git permettant de créer une **version parallèle** de son projet.

C'est un système qui offre la possibilité d'expérimenter de nouvelles fonctionnalités sur une nouvelle branche sans prendre le risque d'altérer la version stable du projet.

La branche par défaut présente dans tous les projets git se nomme **main** et devrait correspondre à la version stable du projet.

Les branches simplifient la collaboration et le développement de nouvelles fonctionnalités.



Utilisation des branches

La commande **git branch** permet de créer, lister, renommer ou supprimer des branches.

Créer une branche

créer une nouvelle branche

git branch <feature>

Supprimer une branche

supprimer une branche

git branch -d <feature>

Renommer une branche

renommer une nouvelle branche

git branch -M <feature> <new-feature>

La commande **git checkout** permet de naviguer dans l'historique sur la même branche ou sur différentes branches.

Utiliser avec l'option -b, elle permet de créer une nouvelle branche.

Changer de branche

Basculer sur une branche

git checkout <feature>

Créer une branche et se déplacer sur celle-ci

Création et bascule

git checkout -b <feature>

Plateformes git



git



GitLab



Bitbucket

Github

Github

Github est une plateforme web, créée en 2008, qui offre un service **d'hébergement** pour des dépôts git.

Elle compte **50 millions d'utilisateurs**, ce qui en fait la plus importante plateforme de dépôt git.

Github propose de nombreuses fonctionnalités facilitant la collaboration et la communication.

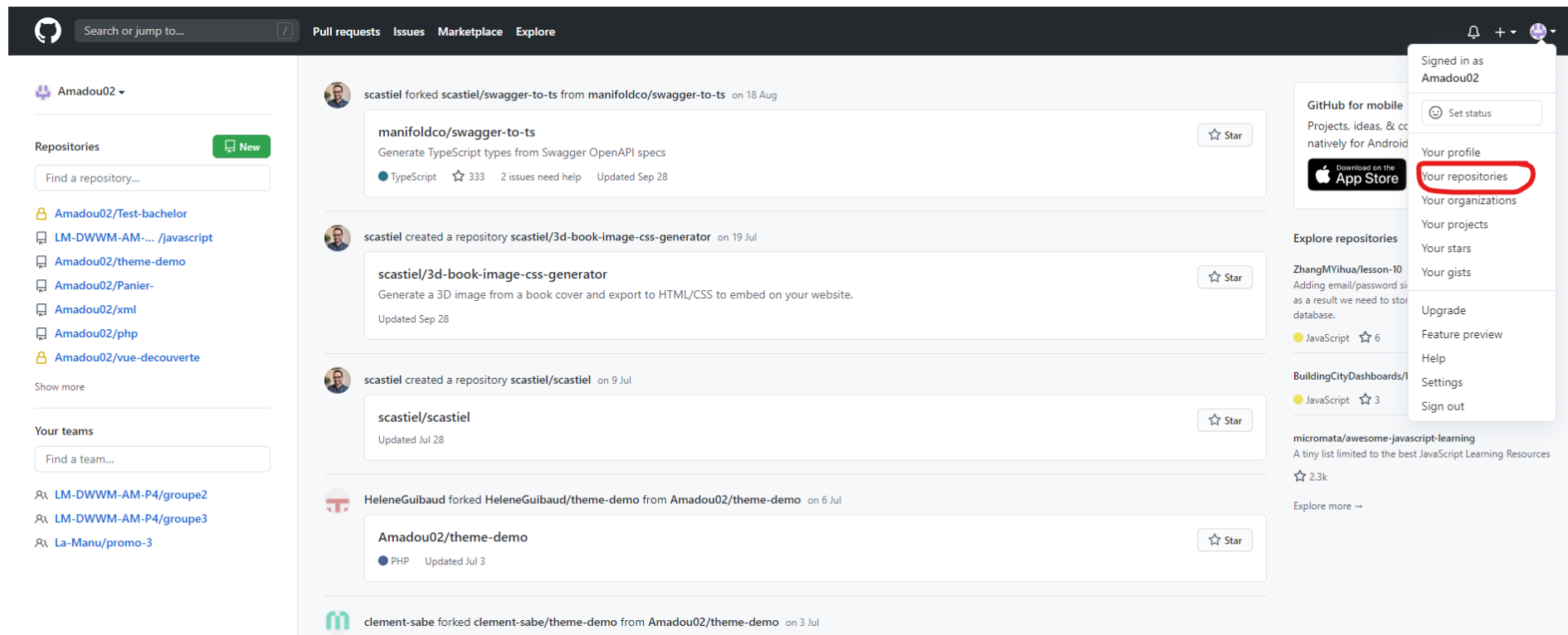
Quelques fonctionnalités disponibles sur github :

- Échange entre développeurs, signalement de bugs dans des projets ouverts.
- Partage de fragments de code (**gists**).
- Contribution à un projet communautaire (**Fork & Pull Request**).
- Publication d'un site web à partir d'un projet (**github pages**)



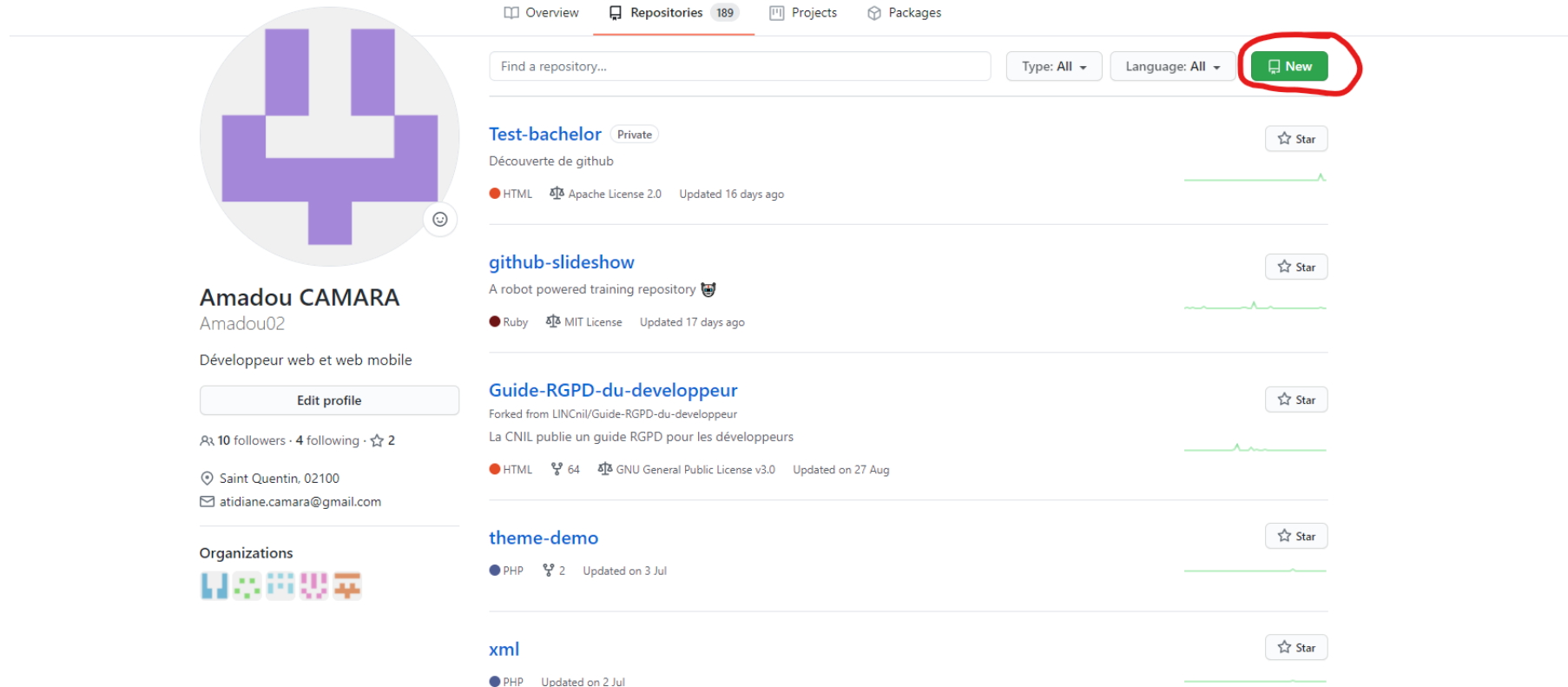
Envoyer son code sur github

Se rendre sur <https://github.com> et cliquer en haut à droite sur votre avatar de profil



Envoyer son code sur github

Cliquez sur New pour créer un nouveau dépôt (Repository)



The screenshot shows the GitHub profile of Amadou CAMARA (Amadou02), a web and mobile developer. The 'Repositories' tab is selected, showing 189 repositories. The 'New' button is highlighted with a red circle. The list of repositories includes:

- Test-bachelor** (Private): Découverte de github, HTML, Apache License 2.0, Updated 16 days ago.
- github-slideshow**: A robot powered training repository, Ruby, MIT License, Updated 17 days ago.
- Guide-RGPD-du-developpeur**: Forked from LINCnil/Guide-RGPD-du-developpeur, La CNIL publie un guide RGPD pour les développeurs, HTML, 64 forks, GNU General Public License v3.0, Updated on 27 Aug.
- theme-demo**: PHP, 2 forks, Updated on 3 Jul.
- xml**: PHP, Updated on 2 Jul.


Envoyer son code sur github

Choisir un nom et une description

Create a new repository

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

Owner *

 Amadou02 ▾

Repository name *

My first project ✓

Great repository names are **My-first-project** but miniature-bassoon?

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.](#)

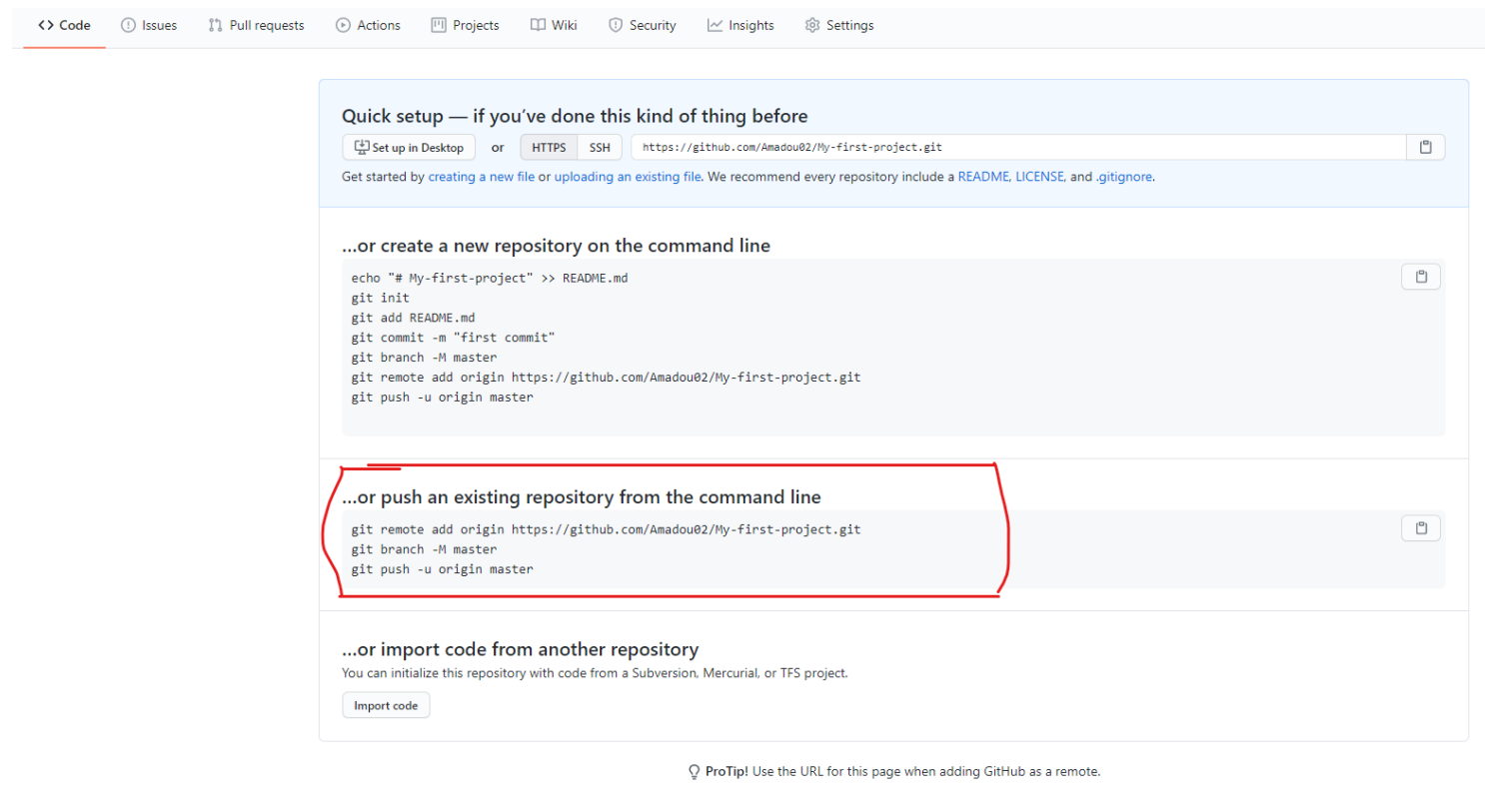
☐ **Choose a license**

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

Create repository

Envoyer son code sur github

Accueil du nouveau repository



<> Code ⓘ Issues 🔗 Pull requests ⚙️ Actions 📁 Projects 📖 Wiki 🛡️ Security 📊 Insights ⚙️ Settings

Quick setup — if you've done this kind of thing before

or

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```

echo "# My-first-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M master
git remote add origin https://github.com/Amadou02/My-first-project.git
git push -u origin master
    
```

...or push an existing repository from the command line

```

git remote add origin https://github.com/Amadou02/My-first-project.git
git branch -M master
git push -u origin master
    
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

💡 **ProTip!** Use the URL for this page when adding GitHub as a remote.

Envoyer son code sur github

Relier le repository github et le dépôt local git et pousser son travail sur github :

Ouvrir **git bash** dans le dossier de votre projet et exécuter ce code

```
git remote add origin https://github.com/Amadou02/My-first-project.git  
git branch -M main  
git push -u origin main
```

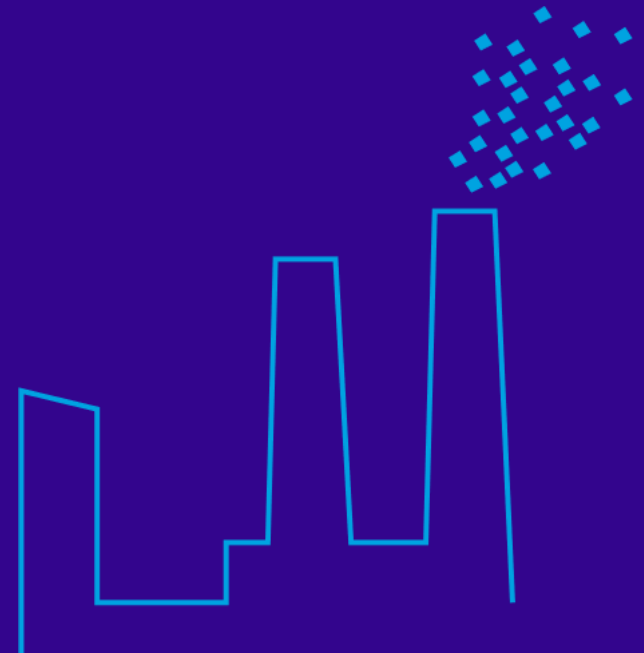
Campus Amiens

`contact-amiens@lamanu.fr`

`https://lamanu.fr`

`70 rue des Jacobins, 80090 Amiens`

`09 86 27 17 04`



Campus Compiègne

contact-compiegne@lamanu.fr

<https://lamanu.fr>

Rue Robert Schuman – La Croix-Saint-Ouen | 60200 Compiègne
Entrée par le 41 rue Irène Joliot Curie (Bâtiment
Millenium II)

09 86 27 17 04



Campus Le Havre

contact-lehavre@lamanu.fr

<https://lamanu.fr>

10 place Léon Meyer, 76600 Le Havre

09 86 27 17 04



Rendez-vous sur lamanu.fr

contact@lamanu.fr

09 86 27 17 04

