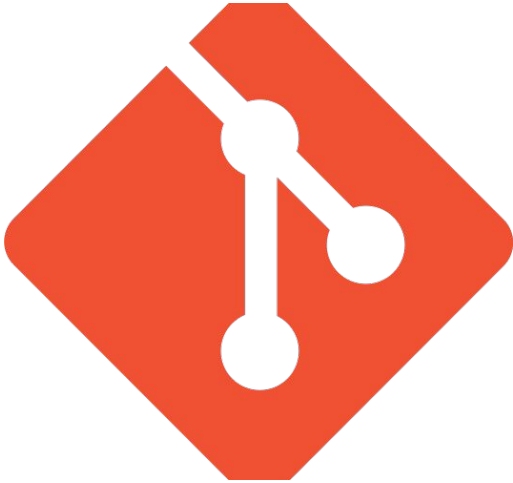




Github-Git



git

Présentation de Git



Git est un logiciel de contrôle de version pour les développeurs.

A quoi cela consiste ?

le contrôle de version c'est le fait d'enregistrer des fichiers tout au long des différentes étapes d'un projet. Cet outil permet de garder une trace de ce qui a été fait dans le projet et de revenir à une version précédente si des modifications sont à faire.

Pourquoi utiliser git ?

git facilite la résolution des erreurs et permet de gérer les erreurs qui pourraient venir pendant le développement

Présentation de GitHub



GitHub permet aux développeurs de stocker et de partager, publiquement ou non, le code qu'ils créent. La plate-forme accueille ainsi, dit-elle, plus de 80 millions de projets, qu'il s'agisse de logiciels, de sites Web, d'applications pour mobile ou tous autres types de programme informatique — et ce quel que soit le langage de programmation utilisé.

Le site est aussi un espace collaboratif. Chaque utilisateur peut contribuer aux projets mis en ligne publiquement sur GitHub, en proposant des modifications.

Pourquoi utilisé GitHub ?

tout simplement simplement pour mettre son travail en ligne et pour voir collaborer avec d'autre développeur sur des projet.

La différence entre GitHub et git



GitHub est une plateforme qui peut contenir des dépôts de code dans [un stockage](#) dans le cloud afin que plusieurs développeurs puissent travailler sur un même projet et voir les modifications des autres en temps réel . De plus GitHub permet d'attribuer des tâches à des individus ou à des groupes mais on peut aussi établir une hiérarchie grâce à des autorisation et les rôles. En outre, les dépôts GitHub sont accessibles au public. Les développeurs du monde entier peuvent interagir et contribuer au code des autres afin de le modifier ou de l'améliorer, ce que l'on appelle le « codage social ».

Git est donc un outils que GitHub utilise pour la collaboration pendant le développement de projet. Il y a trois actions principales que vous pouvez entreprendre lorsqu'il s'agit d'interagir avec le code d'autres développeurs sur Git :

- Fork : Processus consistant à copier le code d'un autre utilisateur du dépôt afin de le modifier.
- Pull : Lorsque vous avez fini de modifier le code de quelqu'un d'autre, vous pouvez le partager avec le propriétaire d'origine via une « pull request ».
- Merge : Les propriétaires peuvent ajouter de nouvelles modifications à leurs projets par le biais d'une fusion, et donner du crédit aux contributeurs qui les ont suggérées.

Résumer

Pour résumer la différence entre **git** et **GitHub** :

1. git est un logiciel VCS local qui permet aux développeurs de sauvegarder des instantanés de leurs projets au fil du temps. Il est généralement mieux adapté à un usage individuel.
2. GitHub est une plateforme web qui intègre les fonctionnalités de contrôle de version de git afin de pouvoir les utiliser en collaboration. Il comprend également des fonctions de gestion de projets et d'équipes, ainsi que des possibilités de mise en réseau et de codage social.

Débuter

Afin d'utiliser ensemble git et GitHub pour le contrôle de version et la collaboration, il y a quelques étapes que vous devrez suivre. Voyons comment ce processus fonctionne.

Il est important de noter que vous devez utiliser git pour profiter de GitHub, vous devrez donc vous familiariser avec le premier avant d'essayer d'intégrer les deux.

Pour Commencer il faudra installer git , l'installation de git varie en fonction de votre système d'exploitation.

Sous **Linux** :

il vous suffit de faire un coup de “sudo apt install git-all”.

Sous Window : vous pouvez aller sur le lien “<https://git-scm.com/download/win>” et le téléchargement démarera automatiquement.

mais vous pouvez aussi télécharger de la source en utilisant l'une des commandes ci dessous :

```
$ sudo dnf install dh-autoreconf curl-devel expat-devel gettext-devel \
```

```
openssl-devel perl-devel zlib-devel
```

```
$ sudo apt-get install dh-autoreconf libcurl4-gnutls-dev libexpat1-dev \
```

```
gettext libz-dev libssl-dev
```

Quelque Termes

une fois l'installation terminé , il y'a certains termes que vous devriez connaître et maîtriser pour utiliser le logiciel Git :

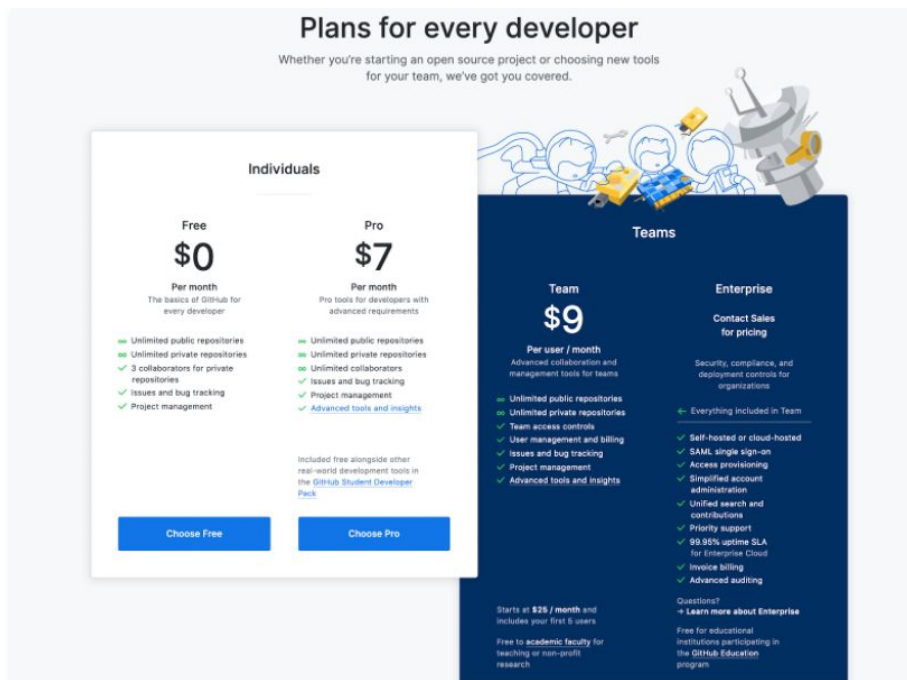
- Repository : L'emplacement du fichier où votre projet est stocké.
- Commit : La commande utilisée pour enregistrer les nouvelles modifications apportées à votre projet dans le dépôt.
- Stage : Avant de pouvoir engager des changements dans Git, vous devez les mettre en stage – cela vous donne la possibilité de préparer votre code avant de l'ajouter officiellement à votre projet.
- Branch : La partie de votre projet que vous développez activement.

Création de compte

Vous aurez donc besoin d'un compte GitHub .

Vous pouvez vous y inscrire gratuitement ou investir dans un forfait payant

via : [GitHub: Where the world builds software · GitHub](https://github.com)



Plans for every developer

Whether you're starting an open source project or choosing new tools for your team, we've got you covered.

Individuals

Free	Pro
\$0	\$7
Per month	Per month
The basics of GitHub for every developer	Pro tools for developers with advanced requirements
<ul style="list-style-type: none">✓ Unlimited public repositories✓ Unlimited private repositories✓ 3 collaborators for private repositories✓ Issues and bug tracking✓ Project management	<ul style="list-style-type: none">✓ Unlimited public repositories✓ Unlimited private repositories✓ Unlimited collaborators✓ Issues and bug tracking✓ Project management✓ Advanced tools and insights
Choose Free	Choose Pro

Included free alongside other real-world development tools in the [GitHub Student Developer Pack](#)

Teams

Team	Enterprise
\$9	Contact Sales for pricing
Per user / month	
Advanced collaboration and management tools for teams	Security, compliance, and deployment controls for organizations
<ul style="list-style-type: none">✓ Unlimited public repositories✓ Unlimited private repositories✓ Team access controls✓ User management and billing✓ Issues and bug tracking✓ Project management✓ Advanced tools and insights	<ul style="list-style-type: none">✓ Everything Included in Team✓ Self-hosted or cloud-hosted✓ SAML single sign-on✓ Access provisioning✓ Simplified account administration✓ Unified search and contributions✓ Priority support✓ 99.95% uptime SLA for Enterprise Cloud✓ Invoice billing✓ Advanced auditing
Starts at \$25 / month and includes your first 5 users	Questions? → Learn more about Enterprise
Free to academic faculty for teaching or non-profit research	Free for educational institutions participating in the GitHub Education program

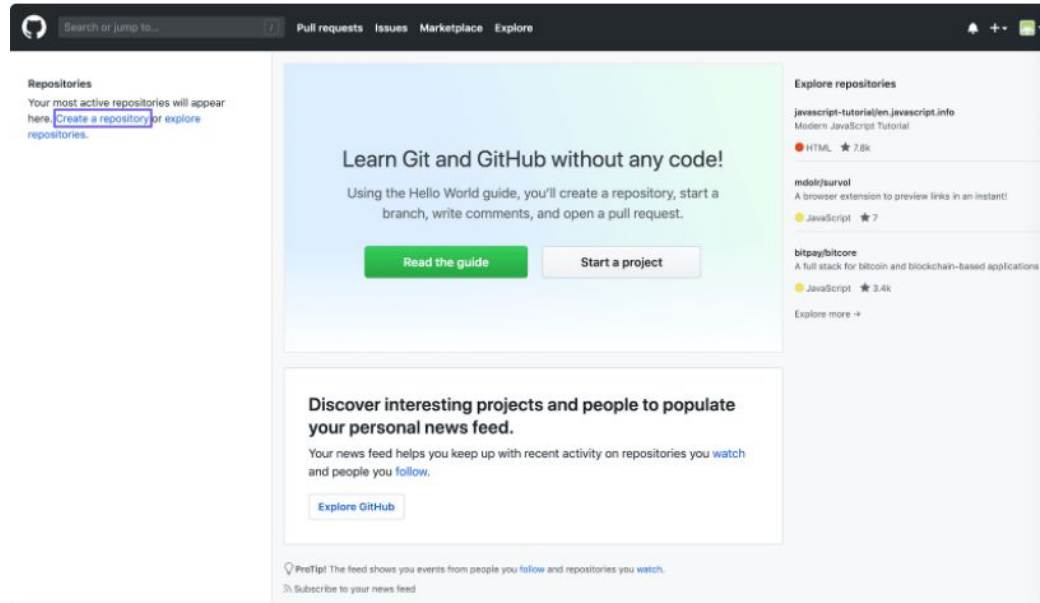
Compte pro et compte gratuit

Un compte gratuit fonctionne bien pour les nouveaux développeurs qui cherchent à affiner leurs compétences.

Un compte pro est mieux adapté aux indépendants et aux codeurs avancés, tandis que les agences voudront investir dans un plan d'équipe afin d'accéder à davantage d'outils de gestion de projet et de communication.

Un repository Github

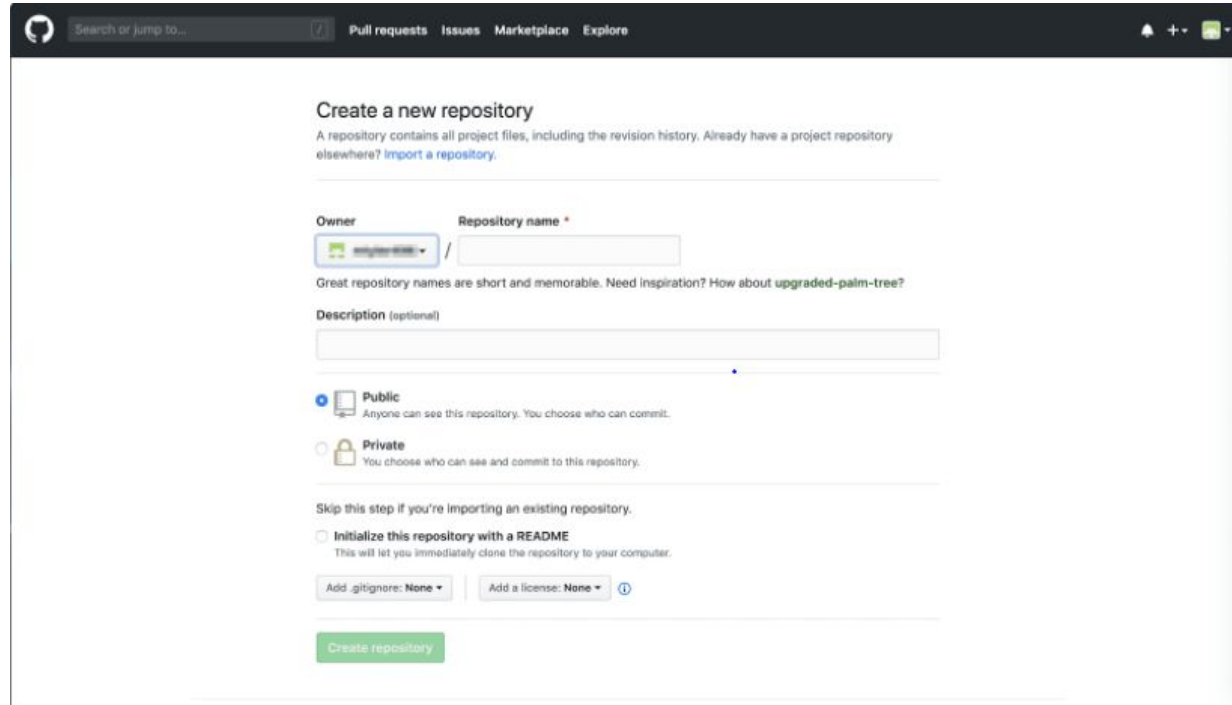
Après avoir créé et configuré votre compte, vous devrez créer un dépôt dans GitHub où vous pourrez stocker votre projet lorsque vous le déplacez depuis git. Vous pouvez le faire en cliquant sur le lien correspondant dans la barre latérale de gauche :



Le dépôt GitHub

Après avoir créer le repository GitHub il faudra donc créer le dépôt comme ci dessous en allant sur create a new repository.

vous pouvez décider de la visibilité de votre dépôt ou non et n'oubliez pas de décocher la case initialize this repository



The screenshot shows the GitHub 'Create a new repository' interface. At the top, there's a navigation bar with a search bar and links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main heading is 'Create a new repository', followed by a subtext: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'

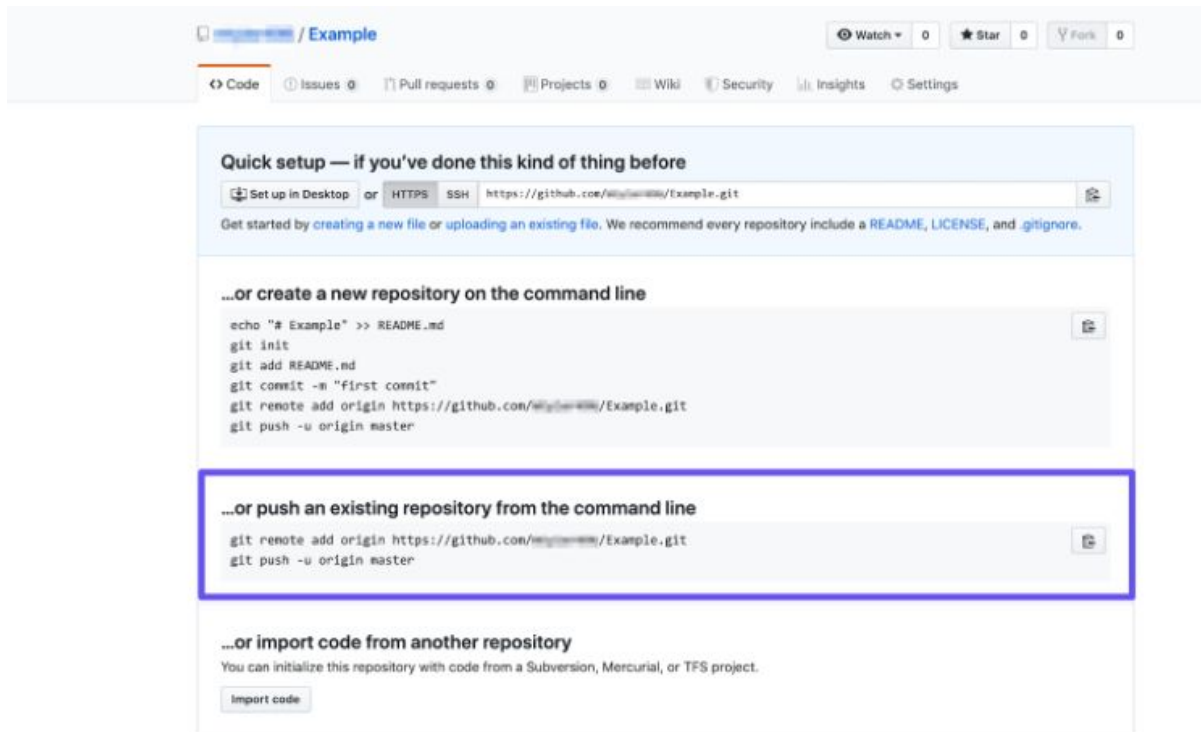
The form includes the following fields and options:

- Owner:** A dropdown menu showing 'mygithubname'.
- Repository name:** An empty text input field.
- Description (optional):** A large text area for a description.
- Visibility:** Two radio button options: 'Public' (selected) and 'Private'. The 'Public' option is described as 'Anyone can see this repository. You choose who can commit.' The 'Private' option is described as 'You choose who can see and commit to this repository.'
- Initialize this repository with a README:** An unchecked checkbox with the text 'This will let you immediately clone the repository to your computer.'
- Additional options:** Two dropdown menus at the bottom: 'Add .gitignore: None' and 'Add a license: None'.

A green 'Create repository' button is located at the bottom of the form.

Push

maintenant vous avez la possibilité d'ajouter du code à votre dépôt grâce au push



Example

Watch 0 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Security Insights Settings

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

Set up in Desktop or HTTPS SSH `https://github.com/Example.git`

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 "# Example" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Example.git
git push -u origin master
```

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

```
git remote add origin https://github.com/Example.git
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.

Import code

Résumé .2

Bien que la différence entre git et GitHub puisse être un peu confuse au début, une fois que vous avez compris les bases, ils sont tous les deux inestimables pour le développement.

Pour intégrer git et GitHub, vous devez suivre les étapes suivantes :

1. Installez git, ajoutez un dépôt et créez un commit.
2. Créez un compte GitHub.
3. Ajoutez un dépôt GitHub à votre compte.
4. “Push” un commit pour GitHub.
5. “Pull” vos changements sur git.

Si vous travaillez avec une équipe pour construire un site web, le contrôle des versions et le partage peuvent être vitaux.

Commande Git

Ci dessous vous aurez la liste des commandes git les plus utilisé sur git .

- **Git config**
- L'une des commandes git les plus utilisées est **git config**. On l'utilise pour configurer les préférences de l'utilisateur : son mail, l'algorithme utilisé pour différencier, le nom d'utilisateur et le format de fichier etc. Par exemple, la commande suivante peut être utilisée pour définir le mail d'un utilisateur: `git config --global user.email jean@google.com`
- **Git init**
- Cette commande est utilisée pour créer un nouveau dépôt GIT : `git init`
- **Git add**
- La **commande git add** peut être utilisée pour ajouter des fichiers à l'index. Par exemple, la commande suivante ajoute un fichier nommé `manger.js` dans le répertoire local de l'index: `git add manger.js`
- **Clone git**
- La **commande git clone** est utilisée pour la vérification des dépôts. Si le dépôt se trouve sur un serveur distant, utilisez: `git clone jean@93.188.160.58:/chemin/vers/dépôt`
- Inversement, si une copie de travail d'un dépôt local doit être créée, utilisez : `git clone /chemin/vers/dépôt`
- **Git commit**
- La **commande git commit** permet de **valider les modifications apportées** au HEAD. Notez que tout commit ne se fera pas dans le dépôt distant : `git commit -m "description du commit"`
- **Git status**
- La **commande git status** affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent encore être ajoutés ou validés. Usage : `git status`
- **Git push**
- **Git push** est une autre commandes GIT de base. Un simple push envoie les modifications locales apportées à la branche principale associée : `git push origin master`
- **Git checkout**
- La **commande git checkout** peut être utilisée pour créer des branches ou pour basculer entre elles. Par exemple nous allons créer une branche : `git checkout -b <nom-branch>`
- Pour passer simplement d'une branche à une autre, utilisez: `git checkout <nom-branch>`