



GitHub Pour les Nuls : Pas de Panique, Lancez-Vous ! (Première Partie)

[xtof](#) 15 décembre 2013 - 3000 mots

Traduction d'un article original de [Lauren Orsini](#) publié le 30 septembre 2013 pour ReadWriteWeb. Seul le [lien original fait référence](#).

La traduction reste à raffiner avec la pratique de cet outil. [Seconde partie en cours d'étude](#) pour me lancer sous peu dans les premiers *commits* à la ligne de commande. Mise en forme prévue pour le plan de route [indieweb](#) 2014. Merci. - [xtof_fr](#)

GitHub est plus qu'un simple outil de programmation. Si vous voulez collaborer sur n'importe quoi, vous devriez l'essayer. 1ère Partie pour apprendre à démarrer sur GitHub.

Nous sommes en 2013 et pas moyen d'y échapper : vous devrez apprendre comment utiliser GitHub.

Pourquoi ? Parce que c'est un réseau social qui change drastiquement notre façon de travailler. Ayant démarré sous forme de plateforme collaborative pour développeurs, GitHub est désormais le plus grand espace de stockage de travaux collaboratifs dans le monde. Que vous soyez intéressé(e) pour participer à ce cerveau global ou tout simplement pour partir à la recherche de cet énorme réservoir de connaissances, vous vous devez d'y être.

En étant simplement membre, vous pourrez croiser le fer avec ce qu'aiment [Google](#) et [Facebook](#). Avant que Github n'existe, les grandes sociétés créaient leurs bases de connaissance surtout en privé. Mais lorsque vous accédez à leurs comptes GitHub, vous êtes libres de télécharger, étudier et construire dessus tout ce que vous voulez sur ce qu'elles ajoutent sur ce réseau. Aussi, qu'attendez-vous ?

Chercher des Réponses GitHub

Aussi gênant que cela puisse paraître, j'ai écrit ce tutoriel parce que je me sentais vraiment perdue dans tous les articles de type "GitHub pour Débutants". Probablement parce qu'à la différence de la plupart des utilisateurs de Github, je manque de bagage solide en programmation. Et je ne m'y retrouvais pas non plus dans les tutoriels d'utilisation de Github, pour construire une vitrine de quelques travaux de programmation.

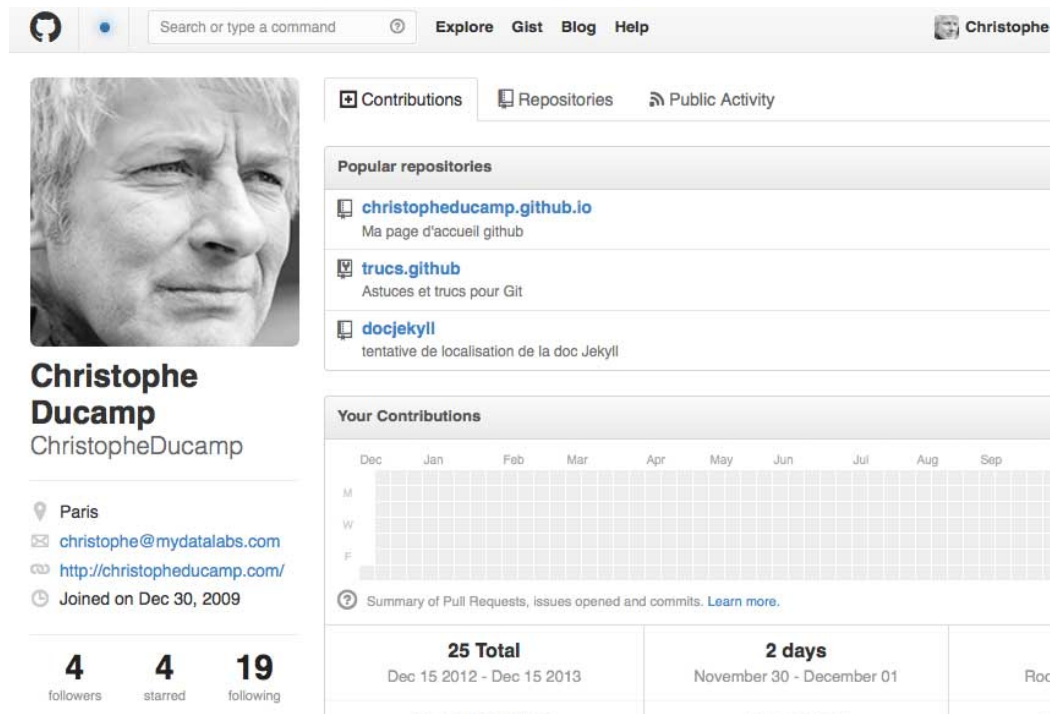
Voir aussi : [Tom Preston-Werner de Github : Comment Nous Sommes Devenus Mainstream](#)

Ce que vous pourriez ignorer, c'est qu'il existe plein de raisons d'utiliser GitHub même si vous n'êtes pas programmeur. Selon les vidéos de tutoriels GitHub, tout travailleur du savoir peut en tirer profit, "knowledge worker" s'entendant ici pour désigner la plupart des professionnels faisant usage d'un ordinateur.

Par conséquent, si vous avez déjà lâché prise sur la compréhension d'utilisation de Github, cet article est pour vous.

L'un des principaux malentendus concernant GitHub est que c'est un outil de développement, faisant partie de la panoplie de tout programmeur, comme le sont les langages de programmation et les compilateurs. Cependant, GitHub en lui-même n'est rien de plus qu'un réseau social comme Facebook ou Flickr. Vous construisez un

profil, vous y déposez des projets à partager et vous vous connectez avec d'autres utilisateurs en suivant leurs comptes. Même si la plupart des utilisateurs y déposent des projets de programmes ou de code, rien ne vous empêche d'y placer des textes ou tout type de fichier à présenter dans vos répertoires de projets.



The screenshot shows the GitHub profile of Christophe Ducamp. The header includes the GitHub logo, a search bar, and navigation links: Explore, Gist, Blog, Help. The user's name 'Christophe' is in the top right. The profile section on the left features a profile picture, the name 'Christophe Ducamp', and the username 'ChristopheDucamp'. Below this, it lists 'Paris' as the location, email 'christophe@mydatalabs.com', website 'http://christopheducamp.com/', and 'Joined on Dec 30, 2009'. At the bottom of the profile, it shows '4 followers', '4 starred', and '19 following'. The main content area has tabs for 'Contributions', 'Repositories', and 'Public Activity'. Under 'Popular repositories', it lists 'christopheducamp.github.io' (Ma page d'accueil github), 'trucs.github' (Astuces et trucs pour Git), and 'docjekyll' (tentative de localisation de la doc Jekyll). The 'Your Contributions' section shows a calendar grid for the year 2013, with a summary of '25 Total' contributions for the period 'Dec 15 2012 - Dec 15 2013' and '2 days' for 'November 30 - December 01'.

Vous avez peut-être déjà plus d'une dizaine de comptes sociaux... et voici pourquoi vous devriez être sur Github : il dispose des meilleures Conditions Générales d'Utilisation. Si vous regardez la section F des [conditions générales](#), vous verrez que Github fait tout pour vous assurer que vous conservez la propriété complète de tous les projets que vous déposez sur le site :

We claim no intellectual property rights over the material you provide to the Service. Your profile and materials uploaded remain yours.

[conditions générales GitHub](#)

En outre, vous pouvez véritablement utiliser GitHub sans connaître

UNE SEULE LIGNE de code. Vous n'avez pas besoin de tutoriel pour vous enregistrer et vous promener. Mais mon point de vue est que GitHub a le mérite de nous apprendre à faire les choses dures en premier, ce qui veut dire, utiliser le bon vieux code Git. Après tout, GitHub est parvenu à produire l'une des interfaces graphiques sans effort pour le langage de programmation Git.

C'est Quoi Git ?

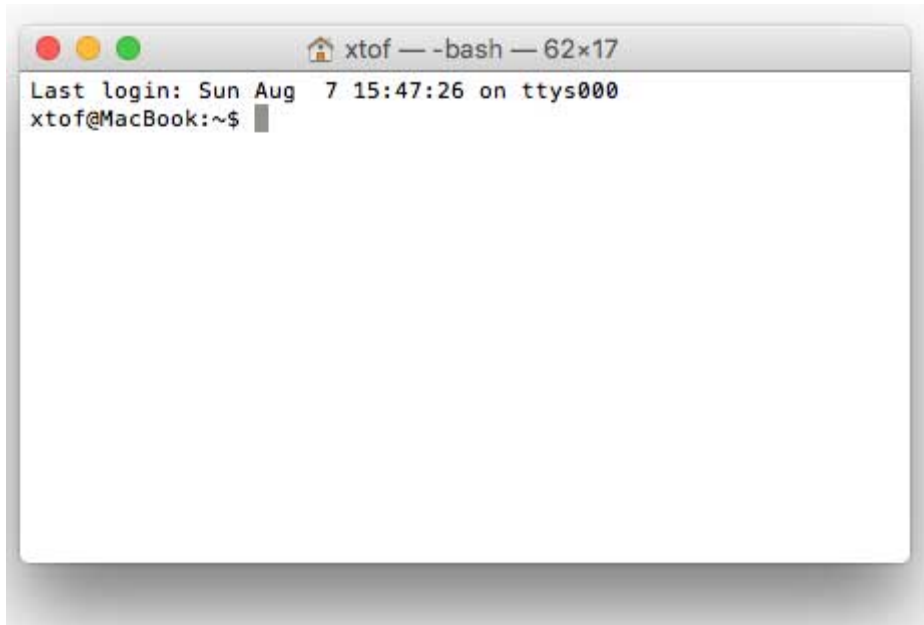
Remercions le célèbre développeur de logiciel [Linus Torvalds](#) pour Git, le logiciel qui fait tourner le coeur de GitHub. (Et tant que vous y êtes, remercions-le aussi pour le système d'exploitation Linux). **Git est un logiciel de contrôle de version**, ce qui signifie qu'il gère les modifications d'un projet sans écraser n'importe quelle partie du projet. Et il ne risque pas de disparaître, tout particulièrement parce que Torvalds et ses collègues développeurs du noyau utilisent Git pour aider à développer le noyau coeur de Linux.

Pourquoi utiliser quelque chose comme Git ? Supposons que vous mettiez à jour avec un collègue des pages sur le même site web. Vous faites des modifications, vous les sauvegardez et les versez sur le site. À ce stade, tout va bien. Le problème survient quand votre collègue travaille sur la même page que vous en même temps. L'un de vous va voir son travail écrasé.

Une [application de contrôle de version](#) comme Git empêche ça d'arriver. Vous et votre collègue pouvez chacun de votre côté verser vos révisions sur la même page, et Git sauvegardera deux copies. Plus tard, vous pourrez fusionner vos modifications sans perdre le travail dans le processus. Vous pouvez même revenir en arrière à tout moment, parce que Git conserve une « copie instantanée » de tous les changements produits.

Le problème avec Git est qu'il est vieux. Si vieux que nous devons utiliser la ligne de commande -ou l'application Terminal si vous êtes

sur Mac - afin d'y accéder, et y taper dedans des bouts de code comme les hackers des années 90. Ceci peut être une proposition difficile pour les utilisateurs d'ordinateurs modernes. C'est là où Github entre dans la danse.



GitHub facilite l'utilisation de Git sur deux points. Premièrement, si vous [téléchargez le logiciel GitHub](#) sur votre ordinateur, GitHub fournit une interface visuelle pour vous aider à gérer localement vos projets avec les contrôles de version. Deuxièmement, créer un compte sur GitHub.com apporte les contrôles de versions à vos projets web, et leur confère des fonctionnalités de réseaux sociaux.

Vous pouvez parcourir les projets d'autres utilisateurs de Github, et même y télécharger des copies pour vous-même afin de les modifier, apprendre ou les enrichir. D'autres utilisateurs peuvent faire la même chose avec vos projets publics, repérer vos erreurs et suggérer des corrections. De toute façon, aucune donnée ne se perd parce que Git enregistre un "instantané" de chaque modification.

Bien qu'il soit possible d'utiliser GitHub sans apprendre Git, il y a une énorme différence entre l'utilisation et la compréhension. Avant de connaître Git, je savais utiliser GitHub, mais je ne comprenais pas

vraiment pourquoi. Dans ce tutoriel, nous allons apprendre à utiliser Git à la ligne de commande.

Les Mots que les Personnes Utilisent quand Elles Parlent de Git

Dans ce tutoriel, il y a quelques mots que j'utiliserai à plusieurs reprises, aucun d'eux dont je n'avais entendu parler avant d'avoir démarré l'apprentissage. Voici les plus connus :

Ligne de Commande : Le programme de l'ordinateur que nous utilisons pour entrer des commandes Git. Sur un Mac, ça s'appelle Terminal. Sur un PC, c'est un programme non-natif que vous téléchargez lorsque vous téléchargez Git pour la première fois (nous allons faire ça dans la section suivante). Dans les deux cas, vous tapez à l'écran des commandes à base de texte, appelées invites de commande, au lieu d'utiliser une souris.

Dépôt : Un répertoire ou de l'espace de stockage où vos projets peuvent vivre. Parfois les utilisateurs GitHub raccourcissent ça en « repo ». Il peut être local sur un répertoire de votre ordinateur, ou ce peut être un espace de stockage sur GitHub ou un autre hébergeur en ligne. À l'intérieur d'un dépôt, Vous pouvez conserver des fichiers de code, des fichiers texte, des images.

Contrôle de Version : Fondamentalement, l'objectif pour lequel Git a été conçu. Quand vous avez un fichier Microsoft Word, vous l'écrasez à chaque fois que vous faites une nouvelle sauvegarde, ou vous sauvegardez plusieurs versions. Avec Git, vous n'êtes plus obligé de faire ça. Git conserve des « instantanés » de chaque point dans l'historique d'un projet, de sorte que vous ne pouvez jamais le perdre ou l'écraser.

Commit : C'est la commande qui donne à Git toute sa puissance. Quand vous « committez », vous prenez un « instantané », une

« photo » de votre dépôt à ce stade, vous donnant un point de contrôle que vous pouvez ensuite réévaluer ou restaurer votre projet à un état précédent.

Branche : Comment plusieurs personnes travaillant sur un projet en même temps sans que Git ne s'embrouille ? Habituellement, elles se « débranchent » du projet principal avec leurs propres versions complètes des modifications qu'elles ont chacune produites de leur côté. Après avoir terminé, il est temps de « fusionner » cette branche pour la ramener vers la branche « master », le répertoire principal du projet.

Commandes Spécifiques Git

Le fait que Git ait été conçu avec un grand projet comme Linux, il existe beaucoup de commandes Git. Toutefois, pour utiliser les bases de Git, vous aurez seulement besoin de connaître quelques termes. Ils commencent tous de la même façon avec le mot « git ».

`git init` : Initialise un nouveau dépôt Git. Jusqu'à ce que vous exécutiez cette commande dans un dépôt ou répertoire, c'est juste un dossier ordinaire. Seulement après avoir entré cette commande, il accepte les commandes Git qui suivent.

`git config` : raccourci de "configurer," ceci est tout particulièrement utile quand vous paramétrez Git pour la première fois.

`git help` : Oublié une commande ? Tapez-ça dans la ligne de commande pour afficher les 21 commandes les plus courantes de Git. Vous pouvez aussi être plus spécifique et saisir "git help init" ou tout autre terme pour voir comment utiliser et configurer une commande spécifique git.

`git status` : Vérifie le statut de votre repository. Voir quels

fichiers sont à l'intérieur, quelles sont les modifications à *commit*, et sur quelle branche du repository vous êtes en train de travailler.

`git add` : Ceci n'ajoute pas de nouveaux fichiers dans votre repository. Au lieu de cela, cela porte de nouveaux fichiers à l'attention de Git. Après avoir ajouté des fichiers, ils sont inclus dans les « instantanés » du dépôt Git.

`git commit` : la commande la plus importante de Git. Après avoir effectué toute sorte de modification, vous entrez ça afin de prendre un "instantané" du dépôt. Généralement cela s'écrit sous la forme `git commit -m "Message ici"`. Le -m indique que la section suivante de la commande devrait être lue comme un message.

`git branch` : Vous travaillez avec plusieurs collaborateurs et vous voulez produire des modifications de votre côté ? Cette commande vous permet de construire une nouvelle branche, ou une chronologie des commits, des modifications et des ajouts de fichiers qui sont complètement les vôtres. Votre titre va après la commande. Si vous vouliez créer une nouvelle branche appelée « chats », vous saisissez `git branch chats`.

`git checkout` : Permet littéralement de vérifier un dépôt dans lequel vous n'êtes pas. C'est une commande de navigation qui vous permet de vous déplacer vers le répertoire que vous voulez vérifier. Vous pouvez utiliser cette commande sous la forme

`git checkout master` pour regarder la branche master, ou `git checkout chats` pour regarder une autre branche.

`git merge` : Lorsque vous avez fini de travailler sur une branche, vous pouvez fusionner vos modifications vers la branche master, qui est visible pour tous les collaborateurs. `git merge chats` prendrait toutes les modifications que vous avez apportées à la branche "cats" et les ajoutera à la la branche master.

`git push` : Si vous travaillez sur votre ordinateur local, et voulez que vos commits soient visibles aussi en ligne sur Github, vous « push »ez les modifications vers Github avec cette commande.

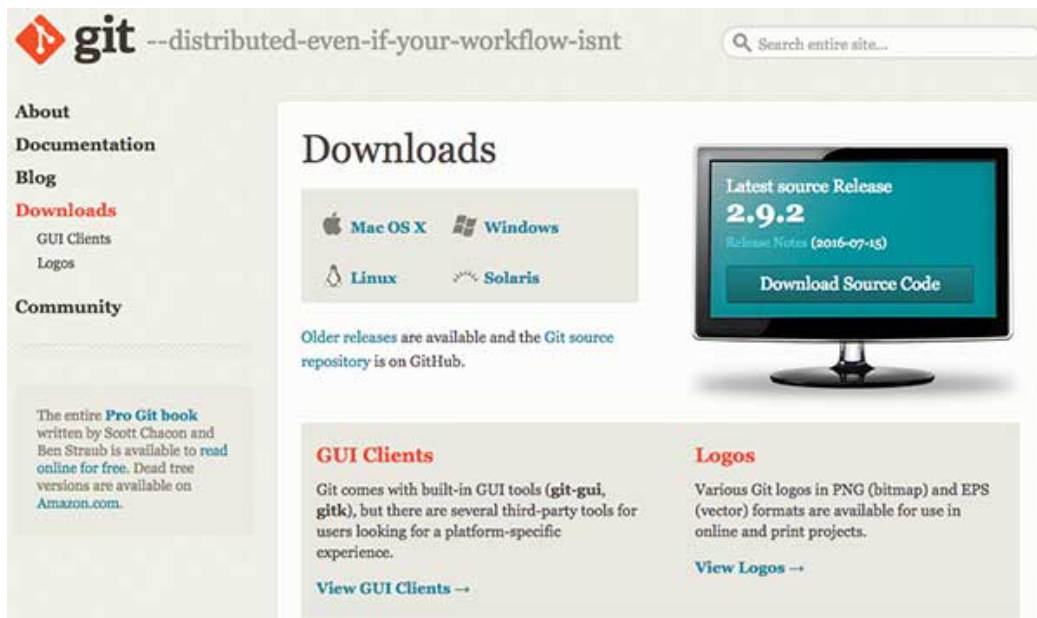
`git pull` : Si vous travaillez sur votre ordinateur local, et que vous voulez la version la plus à jour de votre repository pour travailler dessus, vous “pull”ez (tirez) les modifications provenant de Github avec cette commande.

Paramétrer GitHub ET Git Pour La Première Foix



Premièrement, vous devrez [vous enregistrer](#) pour disposer d'un compte sur GitHub.com. C'est aussi simple que de s'enregistrer sur n'importe quel autre réseau social. Conservez l'e-mail que vous avez choisi à portée de main ; nous en aurons besoin de nouveau.

Vous pourriez vous arrêter là et github fonctionnerait bien. Mais si vous voulez travailler sur votre projet sur votre ordinateur local, vous devez avoir installé Git. En fait, Github ne fonctionnera pas sur votre ordinateur local si vous n'installez pas Git. [Téléchargez et installez la dernière version de Git pour Windows, Mac ou Linux selon votre machine.](#)



Maintenant, il est temps de passer à la ligne de commande. Sur Windows, ça veut dire démarrer l'application Git Bash que vous venez d'installer, et sur MacOSX, c'est le bon vieux Terminal. Il est temps de vous présenter à Git. Saisissez le code qui suit :

```
git config --global user.name "Votre Nom Ici"
```

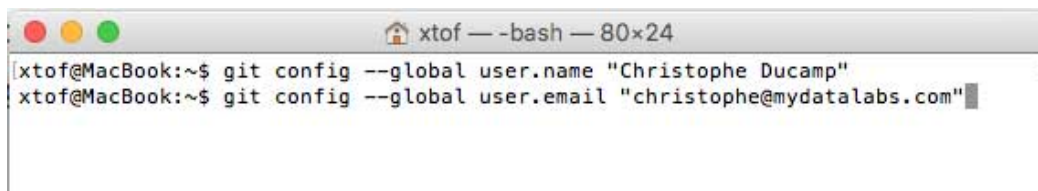
Vous aurez bien sûr besoin de remplacer "Votre Nom Ici" par votre propre nom entre guillemets. Ce peut être votre nom légal, votre pseudo en ligne ou tout ce que vous voudrez. Git s'en moque, il a juste besoin de savoir à qui créditer les commits et projets futurs.

Ensuite, indiquez-lui votre adresse de courrier électronique et assurez-vous que c'est le même email que vous avez utilisé pour enregistrer votre compte Github. Faites comme suit :

```
git config --global user.email "votre_email@votre_
```

C'est tout ce que vous devez faire pour commencer à utiliser Git sur votre ordinateur. Cependant, puisque vous venez de configurer un compte Github.com, il est probable que vous ne souhaitiez pas gérer

simplement votre projet localement, mais aussi en ligne. Si vous le souhaitez, vous pouvez également configurer Git pour qu'il ne vous demande pas de vous connecter à votre compte Github à chaque fois que vous voulez lui parler. Pour les besoins de ce tutoriel, ce n'est pas un grand problème parce que nous ne lui parlerons qu'une fois. Le tutoriel complet pour faire ça est [situé sur Github](#).

A screenshot of a terminal window on a Mac. The window title is 'xtof — -bash — 80x24'. The terminal shows two commands being executed: 'git config --global user.name "Christophe Ducamp"' and 'git config --global user.email "christophe@mydatalabs.com"'. The prompt is 'xtof@MacBook:~\$'.

Créer Votre Repo En Ligne

Maintenant que vous avez tout mis en place, il est temps de créer un endroit pour placer votre projet à faire vivre. Git et Github appellent cela un repository ou « repo » pour faire court, un répertoire numérique ou un espace de stockage où vous pouvez accéder à votre projet, ses fichiers, et toutes les versions de ses fichiers que Git sauvegarde.

Retournons sur GitHub.com et cliquez sur la petite icône de texte à côté de votre nom d'utilisateur. Ou allez vers la nouvelle page repository si toutes les icônes sont les mêmes. Donnez à votre dépôt un nom court et mémorisable. Allez-y et rendez-le public, pourquoi cacher votre tentative d'apprendre Github !

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner

 ChristopheDucamp ▾

Repository name

MonProjet ✓

Great repository names are short and memorable. Need inspiration? How about **automatic-happiness**.

Description (optional)

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾

Add a license: **None** ▾ ⓘ

Create repository

Ne vous inquiétez pas de cliquer sur le bouton radio à côté intitulé "Initialize this repository with a README." Un fichier Readme est généralement un fichier texte qui explique sommairement le projet. Mais nous pouvons produire localement notre propre fichier `Readme` pour l'entraînement.

Cliquez sur le bouton vert "Create Repository" et c'est fait. Vous avez maintenant un espace en ligne pour votre projet restant à faire vivre.

Créer Votre Repository Local

Ainsi nous venons juste de créer un espace pour votre projet en ligne, mais ce n'est pas l'endroit où nous travaillerons dessus. La majeure partie de votre travail sera faite sur votre ordinateur. Nous devons donc en fait refléter ce repository que nous venons juste de

produire sous un répertoire local.

C'est –là où nous faisons quelques saisies de ligne de commande– la partie de chaque tutoriel Git qui me chahute le plus, aussi j'irai vraiment lentement.

Saisissez d'abord :

```
mkdir ~/MonProjet
```

`mkdir` est le raccourci de « make directory ». Ce n'est en réalité pas une ligne de commande Git, mais une commande générale de navigation provenant du temps avant les interfaces ordinateurs visuelles. Le `~/` veille à vous assurer de construire le repository au niveau supérieur de la structure de fichiers de notre ordinateur, au lieu d'un répertoire coincé dans quelque autre répertoire qui serait plus difficile à retrouver. En fait, si vous saisissez `~/` dans la fenêtre de votre navigateur, cela vous ramènera vers le répertoire local le plus haut de votre ordinateur. Pour moi, en utilisant Chrome sur un Mac, cela affiche mon dossier Users.

Remarquez aussi que je l'ai appelé `MonProjet`, le même nom donné au dépôt Github que nous avons produit précédemment. Assurez-vous aussi de garder une cohérence sur votre nom.

Puis, saisissez :

```
cd ~/MonProjet
```

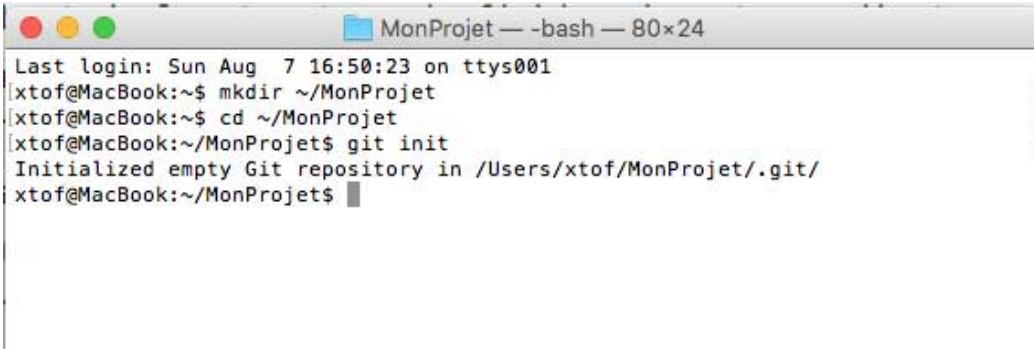
`cd` signifie change directory, et c'est aussi une commande de navigation. Nous venons de produire un répertoire, et nous voulons maintenant passer à ce répertoire et aller dedans. Une fois que nous avons tapé cette commande, nous sommes transportés à l'intérieur

de MonProjet.

Maintenant, nous allons pour finir utiliser une commande Git. Pour votre prochaine ligne, saisissez :

```
git init
```

Vous savez que vous utilisez une commande Git car elle démarre toujours par `git`. Init signifie « initialiser ». Souvenez-vous comment les deux précédentes commandes que nous avons saisies étaient des termes généraux de ligne de commande ? Quand nous tapons ce code à l'intérieur, cela dit à l'ordinateur de reconnaître ce répertoire comme un dépôt local Git. Si vous ouvrez le répertoire, il ne s'affichera pas différemment, parce que ce nouveau répertoire Git est un fichier caché à l'intérieur du dépôt dédié.



```
MonProjet — -bash — 80x24
Last login: Sun Aug  7 16:50:23 on ttys001
[xtof@MacBook:~$ mkdir ~/MonProjet
[xtof@MacBook:~$ cd ~/MonProjet
[xtof@MacBook:~/MonProjet$ git init
Initialized empty Git repository in /Users/xtof/MonProjet/.git/
[xtof@MacBook:~/MonProjet$
```

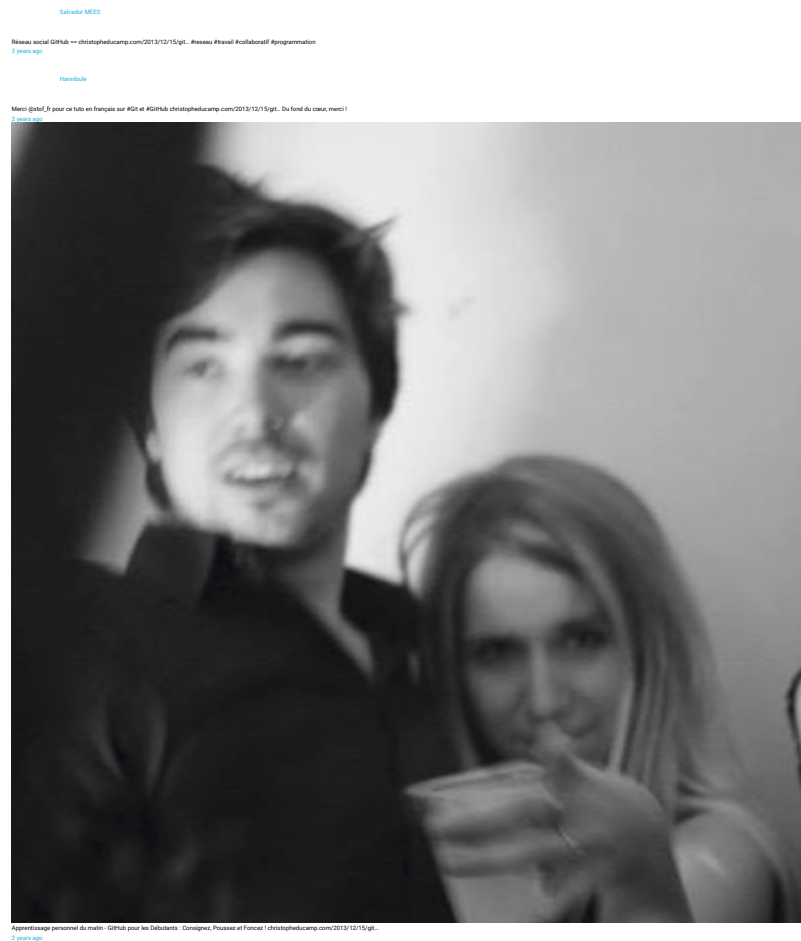
Cependant, votre ordinateur réalise maintenant que ce répertoire est *prêt-pour-git*, et vous pouvez commencer à entrer des commandes Git. Maintenant, vous avez à la fois un dépôt local et un repo en ligne pour votre projet. Dans la [seconde partie](#), vous apprendrez comment faire votre premier commit vers des dépôts locaux et sur Github. Et vous en saurez plus sur quelques ressources géniales de Github.

Voir aussi

- [GitHub pour les Débutants : Consignez, Poussez et Foncez](#)

- [How the Heck Do I Use Github?](#) (Lifehacker Adam Dachis 2013-12-02)

github git tutoriel



↪ ["GitHub Pour les Nuls : Pas de Panique, Lancez-Vous ! \(Première Partie\)"](#) a été mise à jour le : 15 décembre 2013

AMÉLIORER CETTE PAGE

[Email](#) [Twitter](#) [Facebook](#) [GitHub](#) [GitLab](#)

Copyright ©  [xtof](#)