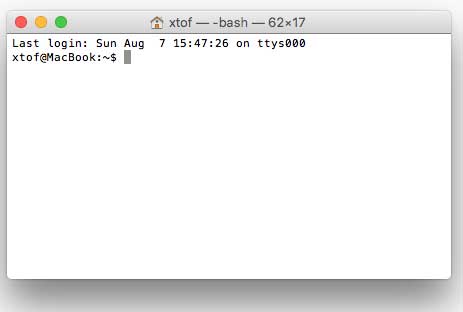
## C’est Quoi Git ?

Remercions le célèbre développeur de logiciel [Linus Torvalds](http://fr.wikipedia.org/wiki/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](http://git-scm.com/video/what-is-version-control) 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](https://desktop.github.com/) 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 allon 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 faire travailler plusieurs personnes 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 seulemenet 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 *commiter*, 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, ceci 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 saisiriez 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 “chats” et les ajoutera à 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.

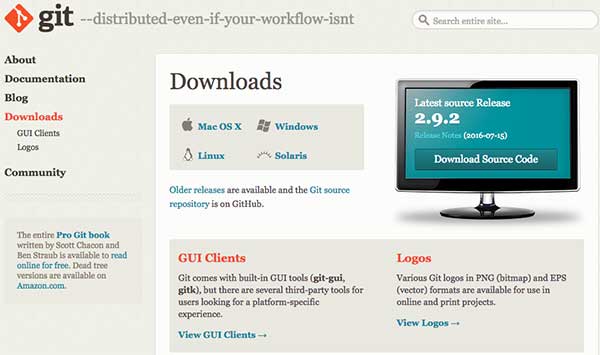
## 

## Paramétrer GitHub ET Git Pour La Première Fois



Premièrement, vous devrez [vous enregistrer](https://github.com/) 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](http://git-scm.com/downloads).



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 :

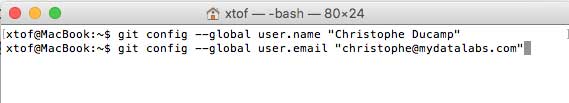
git config --global user.name "Votre Nom Ici"

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\_email.com"

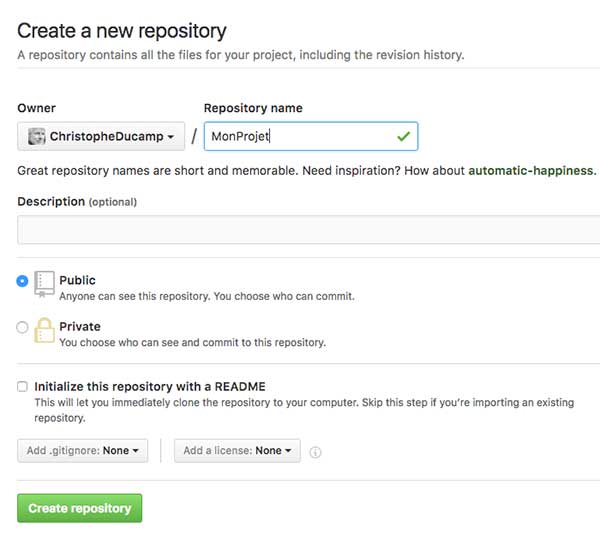
C’est tout ce que vous devez faire pour commencer à utiliser Git sur votre ordinateur. Cependant, puisque vous venez de configurer un compte Github.com, il est probable que vous ne souhaitiez pas gérer simplement votre projet localement, mais aussi en ligne. Si vous le souhaitez, vous pouvez également 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](https://help.github.com/articles/set-up-git).



## 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 !



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 :

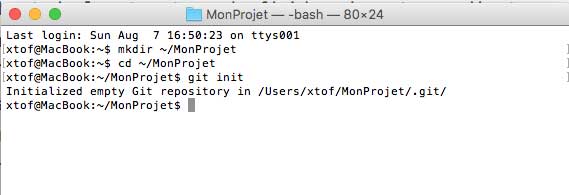
cd ~/MonProjet

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 cettte 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 que 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.



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](https://www.christopheducamp.com/2013/12/16/github-pour-nuls-partie-2/), 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.

SOURCE :

<https://www.christopheducamp.com/2013/12/15/github-pour-nuls-partie-1/>

FIN DU TUTO GITHUB