# **Tuto GIT pour une prise en main rapide !**

## **Introduction**

Les systèmes de contrôle de version aident les développeurs à analyser plus facilement les modifications et les contributions apportées à du code collaboratif. Un VCS est un élément essentiel d’un système de gestion de développement logiciel. Les modifications / révisions / mises à jour effectuées sont identifiables avec des lettres ou des chiffres. Des informations comme la date de modification et l’identité du modificateur sont également renseignées. Dans ce tutoriel, nous vous apprendrons à utiliser un des système de gestion les plus connus : GIT. Vous apprendrez à l’installer et à l’utiliser.

## **Qu’est-ce que GIT?**

En 2005, Linus Torvalds (l’homme connu pour la création du noyau Linux OS) a développé GIT et depuis, il est activement maintenu par Junio ​​Hamano, un ingénieur logiciel japonais. Aujourd’hui, GIT est l’un des systèmes de contrôle de version open source les plus célèbres et des millions de projets à travers le monde s’appuient sur GIT pour la gestion de ceux ci (cela inclut des projets commerciaux et open-source). GIT est un logiciel entièrement gratuit et peut être téléchargé pour Mac, Linux, Windows et Solaris depuis **le site officiel.** Certaines des caractéristiques de GIT:

* Un système de contrôle de version distribué, GIT a une approche **peer to peer** contrairement à d’autres tels que Subversion (SVN) qui optent pour l’approche **client-serveur** .
* GIT permet aux développeurs d’avoir une pléthore de branches de code indépendantes. La création, la suppression et la fusion de ces branches est transparente et prend peu de temps.
* Dans GIT, toutes les opérations sont atomiques; Cela signifie qu’une action peut soit réussir soit échouer (sans aucune altération). Cela est important parce que dans certains systèmes de contrôle de version (comme CVS) où les opérations sont non-atomiques, si une opération dans le dépôt est laissée en suspens, elle peut laisser le dépôt dans un état instable.
* Dans GIT, tout est stocké dans le dossier .git. Ce n’est pas la même chose dans d’autres VCS comme SVN et CVS où les métadonnées des fichiers sont stockées dans des dossiers cachés (par exemple .cvs, .svn, etc.)
* GIT utilise un modèle de données qui aide à garantir l’intégrité cryptographique de tout ce qui est présent dans un dépôt. Chaque fois qu’un fichier est ajouté ou qu’une validation est effectuée, des sums de contrôle sont générés. Aussi, ils sont récupérés via leurs sums de contrôle.
* Une autre excellente caractéristique présente dans GIT est sa **zone de** classement ou **index**. Dans l’index, les développeurs peuvent formater les modifications et les faire réviser avant de les appliquer réellement.

GIT est très simple à utiliser. Pour commencer, vous pouvez soit créer un dépôt soit en rejoindre un. Après l’installation, un simple **git-init** mettra tout ce qu’il faut en place. **git clone** vous permettra de créer une copie du dépôt pour le modifier en local.

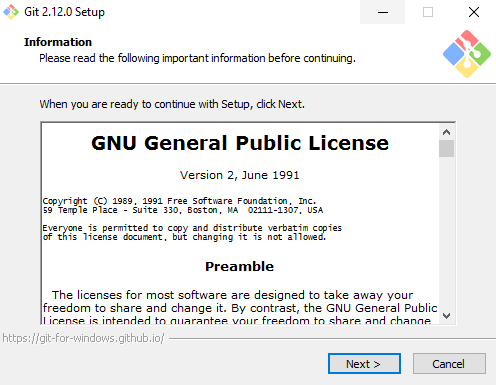
## **Étape 1 – Installation de GIT sur différents systèmes**

Voici les manières les plus simples d’installer GIT:

### **Option 1 – Installation de GIT sous Windows:**

L’installation de GIT sous Windows OS est devenue aussi simple que de télécharger un programme d’installation et de l’exécuter. Procédez comme suit pour configurer GIT sur une machine Windows:

* Visitez **ce site Web** et téléchargez l’installateur GIT pour Windows.
* Une fois téléchargé, double-cliquez sur l’exécutable pour lancer l’assistant d’installation. Il suffit de suivre les instructions à l’écran, de continuer à cliquer sur **Suivant** et enfin T**erminer** pour terminer l’installation.

****

* **Ouvrez une invite de commande. Dans celles-ci, entrez les commandes suivantes:  
  git config --global user.name "Mon Nom" git config --global user.email "votreemail@votreemail.com"**
* **Note: N’oubliez pas de remplacer Mon Nom et votreemail@votreemail.com par vos propres informations. Toute modification faite ultérieurement sera associée à ces détails.**

**C’est tout ce qu’il vous faut pour installer GIT sur Windows !**

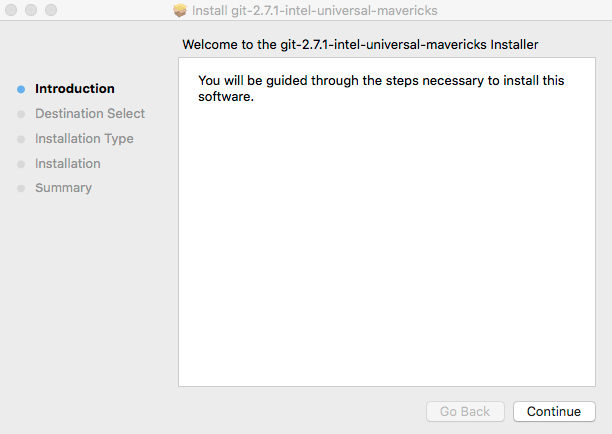
### **Option 2 – Installation de GIT sur MacOS:**

**Il existe plusieurs façons d’installer GIT sur un périphérique Mac. Il y a une chance que GIT soit déjà présent sur votre ordinateur si vous avez XCode d’installé. Exécutez la commande suivante sur un terminal pour vérifier:**

**git --version**

**Si vous obtenez une sortie comme git version 2.7.0 (Apple Git-66) , alors vous avez GIT d’installé. Mais si vous n’avez rien à l’écran, effectuez les étapes suivantes:**

* **Visitez ce site Web et téléchargez le dernier programme d’installation pour Mac.**
* **Suivez les instructions à l’écran et terminez l’installation.**

****

* **Essayez de nouveau la commande git –version afin de confirmer que l’installation a réussi.**
* **Exécutez les commandes suivantes sur un terminal pour configurer votre mail et nom d’utilisateur qui doivent être associés à votre compte GIT:  
  git config --global user.name "Mon Nom" git config --global user.email "votreemail@votreemail.com"**
* **Note: N’oubliez pas de remplacer Mon Nom et votreemail@votreemail.com par vos propres informations. Toute modification faite ultérieurement sera associée à ces détails.**

### **Option 3 – Installation de GIT sous Linux:**

**Si vous êtes un utilisateur Linux, vous devriez être habitué à installer des logiciels et des paquets sur votre ordinateur avec des commandes d’installation telles que apt-get ou yum install. L’installation de GIT est similaire:**

**Pour les utilisateurs Debian / Ubuntu (apt-get):**

* **Ouvrez un terminal et exécutez les commandes suivantes:  
  sudo apt-get update Sudo apt-get install git**
* **Vérifiez que vous avez bien installé GIT avec git –version .**
* **Exécutez les commandes suivantes sur un terminal afin de configurer l’email et le nom d’utilisateur qui doivent être associés à votre compte GIT:  
  git config --global user.name "Mon Nom" git config --global user.email "votreemail@votreemail.com"**
* **Note: N’oubliez pas de remplacer Mon Nom et votreemail@votreemail.com par vos propres informations. Toute modification faite ultérieurement sera associée à ces détails.**

## **Étape 2 – Utilisation de GIT**

**Maintenant que GIT est configuré sur votre périphérique Windows / Mac / Linux, nous allons explorer les bases de GIT. Et comment vous pouvez commencer à utiliser GIT.**

* **Création / configuration / extraction d’un dépôt:**

**Le dépôt est la chose la plus importante d’un projet. Pour transformer n’importe quel dossier en un dépôt GIT, on peut utiliser la commande simple git init <dossier> . Un dossier nommé .git sera créé quand la commande aura été exécutée.**

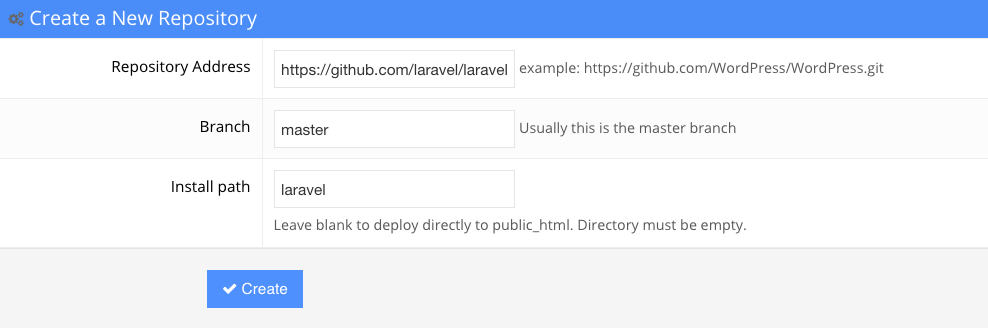
**Inversement, si vous avez déjà un dossier et que vous souhaitez le cloner, vous pouvez utiliser la *commande git clone* . Si vous essayez de consulter un dépôt local, utilisez la commande suivante:**

**git clone /chemin/vers/le/dossier/local**

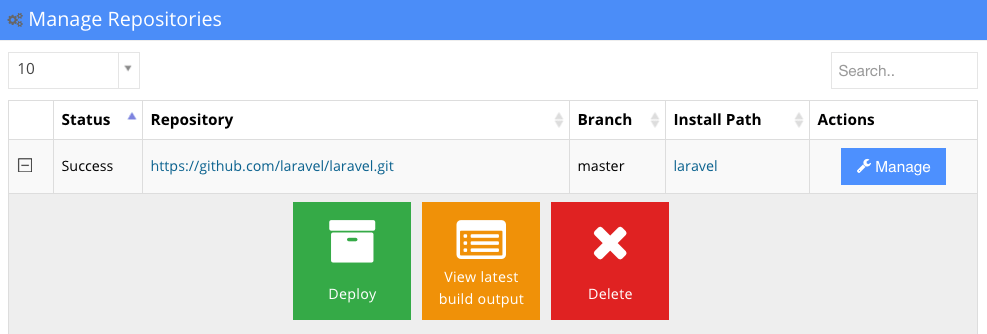
**Si vous avez l’intention de vérifier un dépôt stocké à distance, utilisez**

**git clone utilisateur@hote:/chemin/vers/le/dépôt**

**Si vous avez un compte Hostinger, vous pouvez facilement cloner et gérer des dépôts via la zone Membres -> GIT . Par exemple, si vous voulez cloner un dépôt GIT, entrez simplement son adresse, choisissez une branche, choissisez un chemin et cliquez sur le bouton Créer.**

****

**Une fois la création terminée, vous pourrez gérer votre dépôt dans la même section.**

****

* **Le Workflow:**

**Maintenant qu’un dépôt a été mis en place, parlons de la structure de GIT. Chaque dépôt local se compose de trois “arbres” : le dossier de travail qui contient les fichiers actuels; l’ index qui joue le rôle d’une zone de transit et le HEAD qui est un pointeur vers le dernier commit effectué par l’utilisateur. Voici le workflow : l’utilisateur ajoute un fichier ou des modifications à l’index (la zone de déploiement) et une fois revues, le fichier ou les modifications sont finalement confiées au HEAD .**

* **Les commandes add et commit:**

**Les modifications proposées ou les ajouts de fichiers sont ajoutés à l’index à l’aide de la commande add. Pour ajouter n’importe quel fichier, la commande est:**

**git add <nom\_fichier>**

**Si vous êtes assez confiant pour effectuer ces changements dans votre HEAD , vous pouvez utiliser la commande commit:**

**git commit –m “Message pour décrire le commit”**

**Remarque: Une fois la commande commit exécutée (à partir du dossier de travail), le fichier est affecté au HEAD , mais il n’est toujours pas envoyé au dépôt distant.**

* **Pousser les changements plus loin:**

**Une fois que vous avez validé les modifications (et pensez qu’elles sont prêtes à être envoyées au dépôt d’origine), vous pouvez utiliser la commande push.**

**Une fois que git push origin master est exécuté à partir du dossier de travail, les changements présents dans HEAD sont envoyés au dépôt distant. Dans la commande mentionnée ci-dessus, le master peut être changé par le nom de la branche à laquelle vous souhaitez que les modifications soient effectuées.**

**Cependant, si un dépôt existant n’a pas encore été cloné et que vous souhaitez établir une connexion entre votre dépôt et un serveur distant, procédez comme suit:**

**git remote add origin <serveur>**

**Remarque: Remplacez <serveur> par l’adresse du serveur distant.**

**Une fois cloné, les modifications effectuées seront envoyées au serveur concerné.**

* **Branches:**

**Une autre caractéristique brillante (et avancée) de GIT est la possibilité de créer plusieurs branches indépendantes au sein d’un même projet. Le but principal d’une branche est de développer des fonctionnalités tout en les maintenant isolées l’une de l’autre. La branche par défaut dans tout projet est toujours la branche master. On peut créer autant de branches que nécessaire et finalement fusionner avec la branche master.**

**Une nouvelle branche peut être créée à l’aide de la commande suivante:**

**git checkout -b fonctionnalité\_n \***

**fonctionnalité\_n est le nom de la branche**

**Si vous désirez revenir à la branche master principale, vous pouvez utiliser la commande suivante:**

**git checkout master**

**Toute branche peut être supprimée en utilisant la commande suivante:**

**git checkout -b fonctionnalité\_n**

**Pour rendre la branche disponible aux autres utilisateurs, vous devrez la push vers le dépôt distant; Pour ce faire, utilisez la commande suivante:**

**git push origin fonctionnalité\_n**

* **Mise à jour et fusion:**

**Dans le cas où vous souhaitez mettre à jour votre dossier de travail local à jour, la commande git pull peut être utilisée.**

**Pour fusionner (merge) une autre branche dans celle actuellement active, utilisez: git merge fonctionnalité\_n .**

**Avec push ou merge, GIT essaie toujours de gérer les conflits lui-même, mais parfois il ne peut pas. En cas d’échec en raison de conflits, l’utilisateur doit résoudre les conflits manuellement. Une fois que vous avez édité les fichiers (pour éliminer les conflits), marquez-les comme fusionnés en utilisant:**

**git add <nom.fichier>**

**Si avant de fusionner vous souhaitez afficher les modifications, la commande suivante peut être exécutée:**

**git diff <nom\_de\_branche\_source> <nom\_de\_branche\_cible>**

* **Étiquetage:**

**Avant de faire des mises à jour/modifications, il vous est recommandé de créer des étiquettes. Pour ce faire sur GIT, utilisez la commande suivante:**

**git tag 1.1.0 1c2d2d56fa**

**Le 1c2d2d56fa dans la commande ci-dessus se réfère aux 10 premiers caractères du commit-id référencé avec la balise. L’ID de validation peut être trouvé à partir du journal.**

* **Log:**

**L’historique du dépôt peut être étudié avec le log. La commande git log le récupère. Pour récupérer les commits effectués par un seul utilisateur, vous pouvez utiliser:**

**git log --author =Smith**

**Une version compressée du journal (un commit par ligne) peut être visualisée en utilisant:**

**git log --pretty=oneline**

**Pour afficher uniquement les fichiers qui ont été modifiés:**

**git log --nom-statut**

* **Remplacement des modifications locales:**

**Si vous avez fait une couille et souhaitez revenir sur les modifications apportées à n’importe quel fichier, faites-le en utilisant la commande suivante:**

**git checkout -- <nomfichier>**

**Cela remplacera les changements d’arbre de travail par les dernières données présentes dans le HEAD . Tous les changements qui ont été ajoutés à l’index ne seront pas pris en compte.**

**À l’inverse, si tous les changements / commits locaux doivent être supprimés et que la branche master locale doit pointer vers l’historique le plus récent du serveur, exécutez les commandes suivantes:**

**git fetch origin git reset --hard origin/master**

## **Conclusion**

**Il est important dans un système de développement d’avoir un logiciel qui vous aide à gérer les modifications. Ce tutoriel de base sur GIT devrait permettre à tout développeur d’utiliser facilement GIT. Nous vous rappelons que c’est un système de contrôle de version rigoureux (et très utile) avec tout un tas de fonctionnalités utiles. Si vous avez besoin d’autres informations, la documentation officielle de la GIT pourra vous être utile !**

**J’espère vous avoir été utile ! Nous nous reverrons bientôt pour un futur tutoriel !**

**SOURCE :**

[**https://www.hostinger.fr/tutoriels/tuto-git/**](https://www.hostinger.fr/tutoriels/tuto-git/)

**FIN DU TUTO GIT**