



**Spécialité :**

***Développement web & mobile***

**Module :**

# Programmation web coté serveur

# Chapitre 1 :

Exploiter un outil de travail

Collaboratif

**Git,GitHub**

# Git vs. GitHub : quelle est la différence ?

Pour quelqu'un qui entend parler pour la première fois de *Git*, *GitHub*, le lien apparent peut ne pas être aussi évident.

Git ou GitHub. S'agit-il de la même chose ? Si ce n'est pas le cas, sont-ils liés d'une manière ou d'une autre ?

Ces questions méritent certainement d’être posées. Après tout, Microsoft était prêt à débourser 7,5 *milliards* de dollars pour acquérir GitHub en 2018, les développeurs de tous niveaux devraient donc s’en réjouir. Mais la vérité est que Git et GitHub sont bien plus étroitement liés que Java et JavaScript, mais avec quelques différences clés qui les distinguent.

Quelle est la différence entre Git et GitHub ? Pour répondre à cette question, nous allons examiner chacun d'eux de plus près. Mais avant cela, discutons d'abord du concept de *contrôle de version*.

# Contrôle de version


Les projets de développement ne naissent; ils sont construits ligne par ligne de code à partir de zéro. Et souvent, il faut beaucoup d'essais, d'erreurs et de corrections pour créer quelque chose qui fonctionne réellement comme prévu. C'est là qu'intervient le contrôle de version.

Le contrôle de version est comme un programme d'économies pour votre projet. En suivant et en enregistrant les modifications que vous apportez à votre fichier ou à vos ensembles de fichiers au fil du temps, un système de contrôle de version vous donne la possibilité de réviser ou même de restaurer des versions antérieures. Le contrôle de version prend des instantanés de chaque révision de votre projet. Vous pouvez ensuite accéder à ces versions pour les comparer ou les restaurer selon vos besoins.

Par exemple, imaginons que vous travaillez sur un projet de développement Web et que, au cours de vos révisions, vous remarquez

soudainement que tout votre texte est désaligné. Et comme la première règle du codage est que c'est *toujours* de votre faute, vous pouvez parier qu'une modification que vous avez apportée en cours de route en est la cause. Mais ne vous inquiétez pas ; au lieu de devoir parcourir chaque ligne de code, vous pouvez simplement utiliser votre système de contrôle de version pour recharger les versions précédentes, jusqu'à ce que vous identifiiez la modification en cause et la corrigiez.

Et avec cela à l’esprit, passons à Git.

# Qu'est-ce que Git ?

Développé pour la première fois en 2005, Git est un système de contrôle de version extrêmement populaire qui est au cœur d'une grande variété de projets de grande envergure. Git est installé et maintenu sur votre système local (plutôt que dans le cloud) et vous offre un enregistrement autonome de vos versions de programmation en cours. Il peut être utilisé de manière totalement indépendante de tout service d'hébergement cloud : vous n'avez même pas besoin d'un accès Internet, sauf pour le télécharger.

Comparé à d’autres systèmes de contrôle de version, Git est réactif, facile à utiliser et peu coûteux (g*ratuit*, en fait). Git est également spécialement conçu pour fonctionner correctement avec des fichiers texte. Mais une chose qui distingue vraiment Git est son modèle *de ramification(التشعب)*. La ramification vous permet de créer des branches locales indépendantes dans votre code. Cela signifie que vous pouvez essayer de nouvelles idées, mettre de côté des branches pour le travail de production, revenir à des branches antérieures et supprimer, fusionner et rappeler facilement des branches en un clic.

Et c'est tout. Git est un système de contrôle de version de haute qualité.

# Qu'est-ce que GitHub ?

Il s'agit d'une base de données en ligne qui vous permet de suivre et de partager vos projets de contrôle de version Git en dehors de votre ordinateur/serveur local. Contrairement à Git, GitHub est exclusivement basé sur le cloud. De plus, contrairement à Git, GitHub est un service à but lucratif (bien que les fonctionnalités d'hébergement de référentiel de base soient disponibles gratuitement pour ceux qui souhaitent créer un profil utilisateur, ce qui fait de GitHub un choix populaire pour les projets open source).

En effet, en plus d'offrir toutes les fonctionnalités et avantages de Git, GitHub étend les fonctionnalités de base de Git. Il présente une interface utilisateur extrêmement intuitive, représentée graphiquement, et fournit aux programmeurs des outils de contrôle et de gestion des tâches intégrés. Des fonctionnalités supplémentaires peuvent être implémentées via le service GitHub Marketplace. Et comme GitHub est basé sur le cloud, les référentiels Git d'un individu peuvent être consultés à distance par toute personne autorisée, depuis n'importe quel ordinateur, n'importe où dans le monde (à condition qu'il dispose d'une connexion Internet).

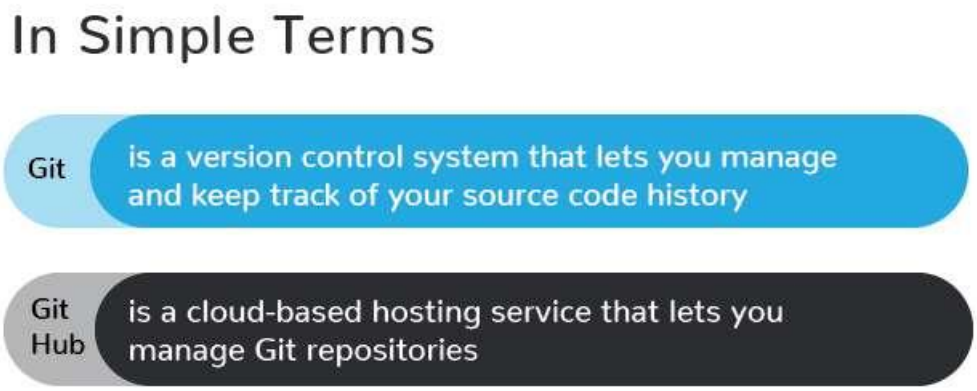
Grâce à GitHub, vous pouvez partager votre code avec d'autres personnes, leur donnant ainsi le pouvoir d'effectuer des révisions ou des modifications sur vos différentes branches Git. Cela permet à des équipes entières de se coordonner sur des projets uniques en temps réel. Au fur et à mesure que des modifications sont introduites, de nouvelles branches sont créées, ce qui permet à l'équipe de continuer à réviser le code sans écraser le travail

des autres. Ces branches sont comme des copies, et les modifications qui y sont apportées ne se reflètent pas dans les répertoires principaux des machines des autres utilisateurs, à moins que ces derniers ne choisissent de pousser/tirer les modifications pour les incorporer. Il existe également une application de bureau GitHub disponible, qui offre des fonctionnalités supplémentaires pour les développeurs expérimentés.

D'autres services d'hébergement de référentiels Git existent également ; GitLab, BitBucket et SourceForge sont tous des alternatives viables à GitHub, et GitLab propose même une option intégrée qui permet aux utilisateurs de GitHub de migrer leurs projets directement vers GitLab.

A lire aussi : Qu'est-ce que GitHub et comment l'utiliser ?

# Git vs. GitHub en termes simples



Alors, dans l'ensemble : Git vs. GitHub… quelle est la différence ? En termes simples, Git est un système de contrôle de version qui vous permet de gérer et de suivre l'historique de votre code source. GitHub est un service d'hébergement basé sur le cloud qui vous permet de gérer les référentiels Git. Si vous avez des projets open source qui utilisent Git, GitHub est conçu pour vous aider à mieux les gérer.

Après tout, dans le monde de la programmation, les conventions de nommage ne sont pas toujours intuitives. C'est pourquoi il est utile de reconnaître les liens et les différences entre les noms similaires de *Git* et *GitHub* . Git et GitHub offrent tous deux aux programmeurs des fonctionnalités de contrôle de version précieuses afin qu'ils puissent créer des projets de codage continus sans avoir peur de tout gâcher. GitHub va juste un peu plus loin que Git, en offrant plus de fonctionnalités et de ressources, ainsi qu'un espace en ligne pour stocker et collaborer sur des projets.

# Installation de Git

## Installation de Git

Avant de commencer à utiliser Git, il faut qu'il soit disponible sur votre ordinateur. Même s'il est déjà installé, c'est probablement une bonne idée d'utiliser la dernière version disponible. Vous pouvez l'installer soit comme paquet ou avec un installateur, soit en téléchargeant le code et en le compilant par vous-même.

| | |
|---|---|
| **Note** | Ce livre a été écrit en utilisant Git version **2.8.0**. Bien que la plupart des commandes utilisées fonctionnent vraisemblablement encore avec d'anciennes version de Git, certaines peuvent agir différemment. Comme Git est particulièrement excellent pour préserver les compatibilités amont, toute version supérieure à 2.8 devrait fonctionner sans différence. |

### Installation sur Linux

Si vous voulez installer les outils basiques de Git sur Linux via un installateur binaire, vous pouvez généralement le faire au moyen de l'outil de gestion de paquet fourni avec votre distribution. Sur Fedora (ou toute distribution parente basée sur RPM, telle que RHEL ou CentOS), vous pouvez utiliser `dnf` :

```
$ sudo dnf install git-all
```

Sur une distribution basée sur Debian, telle que Ubuntu, essayez `apt` :

```
$ sudo apt install git-all
```

Pour plus d'options, des instructions d'installation sur différentes versions Unix sont disponibles sur le site web de Git, à https://git-scm.com/download/linux.

### Installation sur macOS

Il existe plusieurs méthodes d'installation de Git sur un Mac. La plus facile est probablement d'installer les **Xcode Command Line Tools**. Sur Mavericks (10.9) ou postérieur, vous pouvez simplement essayer de lancer `git` dans le terminal la première fois.

```
$ git --version
```

S'il n'est pas déjà installé, il vous demandera de le faire.

Si vous souhaitez une version plus à jour, vous pouvez aussi l'installer à partir de l'installateur binaire. Un installateur de Git pour macOS est maintenu et disponible au téléchargement sur le site web de Git à https://git-scm.com/download/mac.

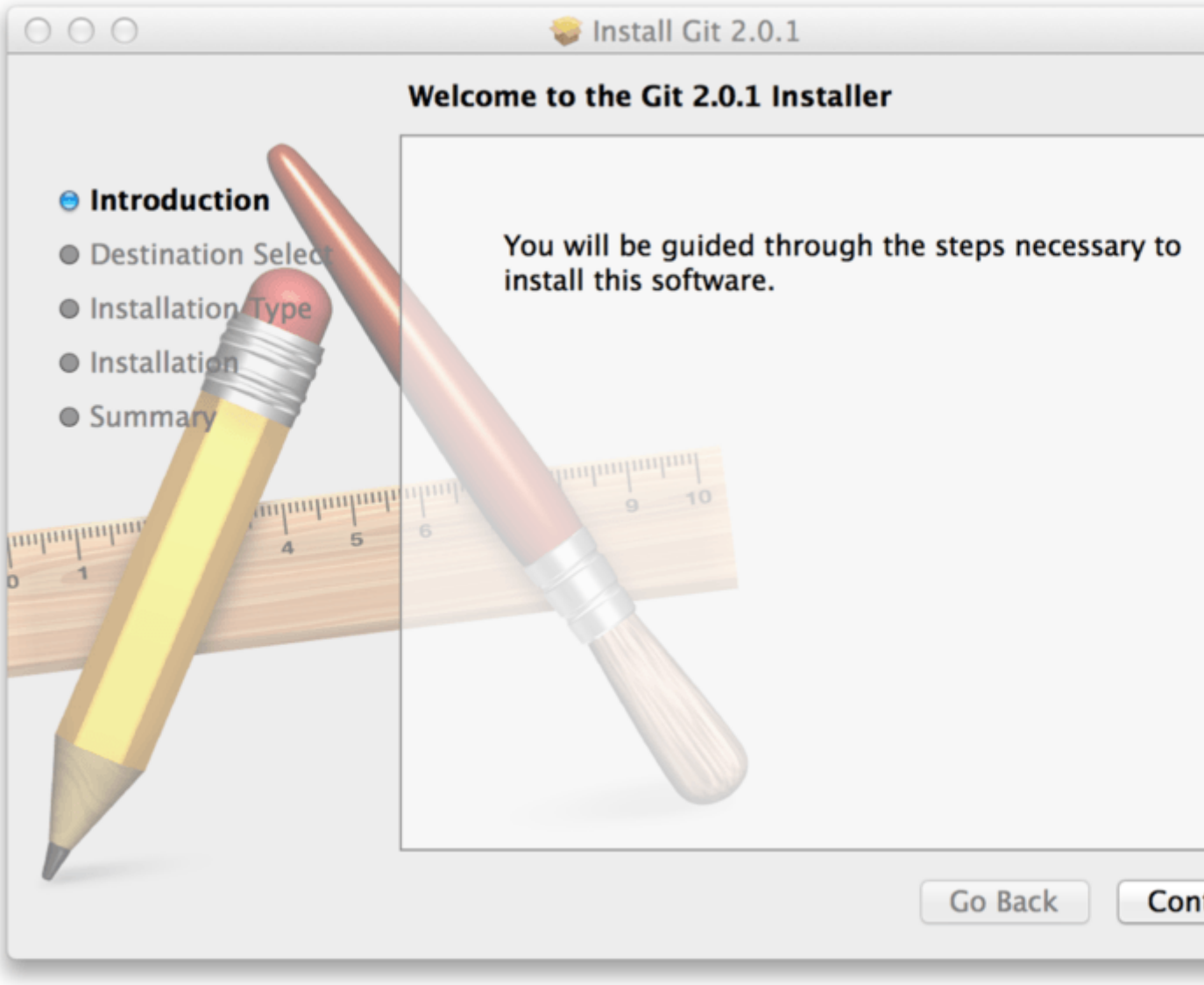



Figure 7. Installateur de Git pour macOS

Vous pouvez aussi l'installer comme sous-partie de l'installation de GitHub pour macOS. Leur outil Git graphique a une option pour installer les outils en ligne de commande. Vous pouvez télécharger cet outil depuis le site web de GitHub pour macOS, à https://desktop.github.com.

### Installation sur Windows

Il existe aussi plusieurs manières d'installer Git sur Windows. L'application officielle est disponible au téléchargement sur le site web de Git. Rendez-vous sur https://git-scm.com/download/win et le téléchargement démarrera automatiquement. Notez que c'est un projet nommé **Git for Windows** (appelé aussi msysGit), qui est séparé de Git lui-même ; pour plus d'information, rendez-vous à https://msysgit.github.io/.

Pour obtenir une installation automatisée, vous pouvez utiliser le paquet Chocolatey Git. Notez que le paquet Chocolatey est maintenu par la communauté.

Une autre méthode facile pour installer Git est d'installer **Github for Windows**. L'installateur inclut une version en ligne de commande avec l'interface graphique. Elle fonctionne aussi avec PowerShell et paramètre correctement les caches d'authentification et les réglages CRLF. Nous en apprendrons plus sur ces sujets plus tard, mais il suffit de savoir que ces options sont très utiles. Vous pouvez télécharger ceci depuis le site de **Github for Windows**, à l'adresse https://windows.github.com.

### Installation depuis les sources

Certains peuvent plutôt trouver utile d'installer Git depuis les sources car on obtient la version la plus récente. Les installateurs de version binaire tendent à être un peu en retard, même si Git a gagné en maturité ces dernières années, ce qui limite les évolutions.

Pour installer Git, vous avez besoin des bibliothèques suivantes : autotools, curl, zlib, openssl, expat, libiconv. Par exemple, si vous avez un système d'exploitation qui utilise dnf (tel que Fedora) ou apt-get (tel qu'un système basé sur Debian), vous pouvez utiliser l'une des commandes suivantes pour installer les dépendances minimales pour compiler et installer les binaires Git :

```
$ 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
```

Pour pouvoir ajouter la documentation dans différents formats (doc, html, info), ces dépendances suppplémentaires sont nécessaires :

```
$ sudo dnf install asciidoc xmlto docbook2X

$ sudo apt-get install asciidoc xmlto docbook2x
```

| | |
|---|---|
| **Note** | Les utilisateurs de RHEL ou dérivés tel que CentOS et Scientific Linux devront activer le dépôt EPEL pour télécharger le paquet `docbooc2X`. |

Si vous utilisez une distribution basée sur Debian (Debian/Ubuntu/dérivés d'Ubuntu), vous avez aussi besoin du paquet `install-info` :

```
$ sudo apt-get install install-info
```

Si vous utilisez une distribution basée sur RPM (Fedora/RHEL/dérivés de RHEL), vous avez aussi besoin du paquet `getopt` (qui est déjà installé sur les distributions basées sur Debian) :

```
$ sudo dnf install getopt
```

De plus, si vous utilisez Fedora/RHEL/dérivé de RHEL, vous devez faire ceci :

```
$ sudo ln -s /usr/bin/db2x_docbook2texi /usr/bin/docbook2x-texi
```

à cause des différences de nom des binaires.

Quand vous avez toutes les dépendances nécessaires, vous pouvez poursuivre et télécharger la dernière version de Git depuis plusieurs sites. Vous pouvez l'obtenir via Kernel.org, à https://www.kernel.org/pub/software/scm/git, ou sur le miroir sur le site web GitHub à https://github.com/git/git/releases. L'indication de la dernière version est généralement plus claire sur la page GitHub, mais la page kernel.org a également des signatures de version si vous voulez vérifier votre téléchargement.

Puis, compilez et installez :

```
$ tar -zxf git-2.8.0.tar.gz

$ cd git-2.8.0

$ make configure

$ ./configure --prefix=/usr

$ make all doc info

$ sudo make install install-doc install-html install-info
```

Après ceci, vous pouvez obtenir Git par Git lui-même pour les mises à jour :

```
$ git clone git://git.kernel.org/pub/scm/git/git.git
```

# Les 10 principales commandes Git

## #1 : « statut git »

Cette commande simple fournit un certain nombre de détails sur la branche actuelle. Ces détails incluent l'état de mise à jour de la branche et la présence ou non de quelque chose à valider, à envoyer ou à extraire. Cela montre également toutes les autres modifications qui ont été mises en scène, telles que les fichiers mis en scène ou les fichiers créés/modifiés/supprimés.

## #2 : « git show »

Utilisation : git show [commit]

Cette commande est utilisée pour afficher les métadonnées d'un commit spécifié.

## #3 : « git init »

Utilisation : git init [nom du dépôt]

Il s'agit de la commande permettant de démarrer un nouveau référentiel.

## #4 : « git commit »

Utilisation : git commit -m « [Saisissez le message de validation] »

Utilisez la commande « commit » pour enregistrer une copie du fichier dans l’historique des versions du projet.

## #5 : « branche git »

Utilisation : branche git

Lorsque votre ligne de commande est située dans un référentiel particulier, vous pouvez utiliser « git branch » pour voir une liste de toutes les branches locales de ce référentiel actuel.

## #6 : « git add »

Utilisation : git add [fichier]

Une commande permettant d'ajouter un fichier à la zone de préparation.

# #7 : « git log »

Utilisation : git log

Une méthode simple pour afficher l’historique des versions de la branche actuelle.