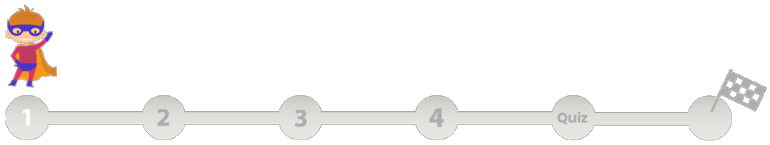
1 - Git, un système de contrôle de version



1. Git, un système de contrôle de version

Git est un **outil puissant**, mais il a la réputation de déconcerter la plupart des nouveaux venus. Avec les bonnes connaissances, n'importe qui peut maîtriser git.

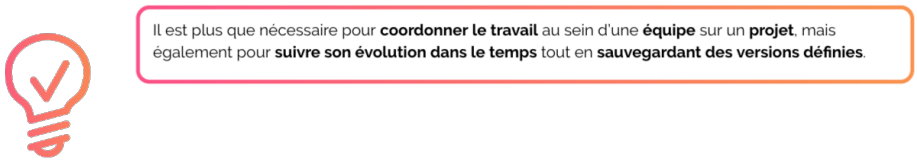
C'est un outil indispensable à la pratique du DevOps en entreprise.
Il permet de maîtriser les versions et les modifications des logiciels conçus par les développeurs, mais également des scripts ou outils utilisés pour le déploiement d'applicatifs.

Figure 1 : Logo de git

1.1 Logiciel de gestion de versions

Tout d'abord, **git est différent de GitHub**. Ce dernier est un site web communautaire utilisé pour l'hébergement de projets qui utilisent git. On pourrait presque le considérer comme un réseau social pour utilisateur de git.

Par exemple, lorsque vous éditez un fichier, git peut vous aider à déterminer exactement ce qui a changé, qui l'a modifié et pourquoi.



Vous pouvez l'utiliser lors de la rédaction d'un essai ou pour suivre les modifications apportées aux fichiers d'illustrations et de conception. Il existe même un projet sur la plateforme Github visant à versionner le code civil : https://github.com/steeve/france.code-civil.

Git n'est **pas le seul système** de contrôle de version, mais c'est de loin le plus populaire, vous trouvez les sources des plus grands projets open sources tels que le noyau Linux, Docker, Kubernetes.
Tout membre d'une équipe DevOps doit savoir utiliser git quotidiennement.

Dans les projets complexes, il arrive fréquemment que plusieurs personnes apportent des modifications aux mêmes fichiers simultanément, il y aura forcément des conflits à résoudre entre utilisateurs.

Heureusement, il existe des outils de **« merge conflict »** qui vous permettront de **résoudre automatiquement la plupart des conflits**.

1.2 Démarrer avec git

Git est installé par défaut sur de nombreux systèmes d'exploitation GNU/Linux récents.

Dans le cas contraire, ou si vous utilisez Windows ou macOS, vous pourrez télécharger git de deux manières :

- en utilisant l'**interface en ligne de commande** (CLI pour Command Line Interface) de git : https://git-scm.com/downloads.
- à travers une **interface utilisateur graphique** (GUI, pour Graphical User Interface) : https://git-scm.com/downloads/guis.
  Il existe une multitude d'interfaces graphiques pour git, si vous ne savez pas laquelle choisir je vous conseille **GitKraken** qui est disponible pour Windows, GNU/Linux et macOS.

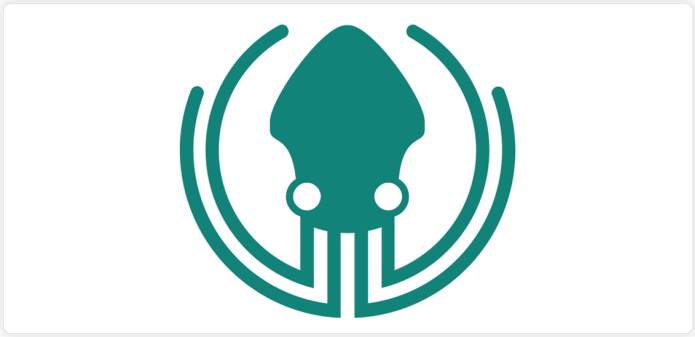


Figure 2 : Logo de GitKraken

Je ne peux que vous conseiller de commencer à utiliser git à travers le CLI, vous arriverez mieux à comprendre les mécaniques du logiciel. Une fois que vous en maitriserez les arcanes, vous pourrez librement choisir la version avec GUI qui vous convient le mieux.

Dans cette fiche nous n'aborderons git qu'à travers le CLI pour vous forcer à le prendre en main. Il faudra donc vous familiariser avec la ligne de commande, l'un des principaux outils du DevOps.

1.3 Le vocabulaire de git

Pour bien maitriser le monde de git, il va falloir dans un premier temps vous familiariser avec le vocabulaire de git. Voici un petit glossaire pour bien débuter :

- **dépôt** : vous trouverez aussi le terme de « répo » ou « repository » en anglais. C'est le dossier ou votre projet sera stocké et versionné. Il peut être sur votre machine en local ou bien sur un serveur distant.
- **commit** : c'est une commande qui va vous permettre de créer une image, ou « snapshot » en anglais, de l'état courant de votre dépôt git. Ces images vous permettront de naviguer à travers l'historique de votre répo.
- **commit hash** : c'est un identifiant unique auto-généré (au format SHA1) lié à votre commit, il permet de retrouver facilement votre commit.
- **branch** : permet de désigner un espace de développement où vous pourrez travailler sans perturber le travail des autres collaborateurs du projet. C'est l'un des principaux fondements de git. Vous devez travailler sur votre branche, une fois votre travail terminé, vous fusionnerez votre branche avec la branche initiale sur laquelle votre branche se base. Vous l'aurez compris, il y a forcément une branche de créée lorsque vous initialisez un dépôt git., cette branche s'appelle la branche « master ».
- **staging area** ou **index** : un cache des fichiers à commiter, une zone ou les fichiers sont indexés avant d'être envoyés sur un repo distant.
- **staged files** : fichiers déjà indexés et prêt à être versionné au prochain commit.
- **unstaged files** : fichiers non indexés et qui ne seront pas versionnés au prochain commit.
- **working directory** : le dossier racine où vous apportez vos modifications.
- **working copy** : la liste des dossiers et fichiers sur lesquels vous êtes en train de travailler localement.
- **snapshot** : c'est une version, figée dans le temps, de votre projet à un instant « t » lors d'un commit.
- **history** : c'est un récapitulatif contenant toutes les modifications de votre projet qui ont été commitées. On peut le voir comme un album photo chronologique listant les snapshots effectués et l'ordre dans lequel ils ont été faits.