Git - Partie 1 - Les bases

1. Histoire de Git

Le noyau Linux est un projet libre de grande envergure. Pour la plus grande partie de sa vie (1991–2002), les modifications étaient transmises sous forme de patchs et d'archives de fichiers. En 2002, le projet du noyau Linux commença à utiliser un DVCS propriétaire appelé BitKeeper.
En 2005, les relations entre la communauté développant le noyau Linux et la société chargée du développement de BitKeeper furent rompues, et le statut de la gratuité de l'outil fut révoqué.
La communauté de dev de Linux et Linus Torvalds décidèrent donc de créer leur propre outil avec comme objectif :

- vitesse
- support pour les développements non linéaires (branches)
- complètement distribué
- capacité à gérer efficacement des projets d'envergure tels que le noyau Linux

Source

2. A quoi ça sert ?

- Les systèmes de contrôle de version considèrent l'information qu'ils gèrent comme une liste de fichiers et les modifications effectuées sur chaque fichier dans le temps. Git pense ses données comme un instantané (photo à l'instant T). A chaque fois que vous validez ou enregistrez l'état du projet dans Git, il prend un instantané du contenu et enregistre une référence à cet instantané. Si les fichiers n'ont pas changé, Git ne stocke pas le fichier à nouveau, juste une référence vers le fichier original qu'il a déjà enregistré. Git pense ses données à la manière d'un flux d'instantanés.

- La plupart des opérations de Git ne nécessitent que des fichiers et ressources locales. Pas besoin d'un autre ordinateur, de réseau ou d'un serveur. Il lit simplement dans votre base de données locale.

- Permet le travail collaboratif, chacun peut travailler sur le même projet sur sa propre branche.

- Intégrité du code source. Tous les éléments sont sécurisés à l'aide d'un algorithme.

- Historique de tous les changements ou modifications. Versionning.

Source Source_2

3. Le SSH

- Qu'est-ce que le SSH ?

  Secure SHell
  C'est un protocole de communication et un programme informatique.
  Il permet la connexion d'une machine distante (serveur) via une liaison sécurisée dans le but de transférer des fichiers ou des commandes en toute sécurité.
  Le SSH impose une authentification des deux côtés de l'échange à l'aide de clés. Une privé et une publique.

  En français : Carapace sécurisée

- Création de clé ssh et ajout à github

  A. Créer la clé ssh

     ```
     ssh-keygen -t rsa -b 4096
     ```

  B. Récupérer la clé publique

     - La clé publique est stockée dans **~/.ssh/id_rsa.pub**. C'est ce fichier qu'il faut transmettre quand on vous demande votre clé publique.
     - La clé privée est stockée dans **~/.ssh/id_rsa**

     ```
     cat ~/.ssh/id_rsa.pub
     ```

  C. Ajouter la clé SSH publique dans github

     - Settings
     - SSH and GPG keys
     - New SSH key

  D. Mettre en mémoire la passphrase

     - Lancer l'agent ssh :
       `eval `ssh-agent -s``

- Ajouter et sauvegarder la passphrase:
  `ssh-add`

- Entrer votre passphrase

4. Configuration de git

- Définir le nom que vous voulez associer à toutes vos opérations de commit:

  `git config --global user.name “mon nom”`

- Définir l'email que vous voulez associer à toutes vos opérations de commit:

  `git config --global user.email “email@email.fr”`

5. Commandes de base

- Télécharge un projet et tout son historique de versions :

  `git clone [url]`

- Ajoute un instantané du fichier, en préparation pour le suivi de version :

  `git add [nom du fichier]`

- Enregistre des instantanés de fichiers de façon permanente dans l'historique des versions :

  `git commit -m “message descriptif”`

- Liste tous les nouveaux fichiers et les fichiers modifiés à commiter :

  `git status`

- Envoie tous les commits de la branche locale vers GitHub:

  `git push [alias] [branche]`

- Récupère tout l'historique du dépôt nommé et incorpore les modifications :

  ```
  git pull
  ```

Aide-mémoire