1 Création d'une clé SSH

La clé SSH permet d'accéder à GitHub via git. Elle permet notament de ne pas avoir à s'identifier lorsque l'on envoie des commits (modification) à gitHub. Je liste les commandes mais toute la doc est disponible sur GitHub en suivant le lien générer une clé SSH et ajouter à un agent

Génération d'une nouvelle clé GPG

- Ouvrez Terminal.
- Collez le texte ci-dessous (apres le $) en indiquant l'adresse e-mail GitHub. (Rq : le ed25519 est important ne pas le changer, c'est un algorithme de communication plus avancé que RSA)
  - $ **ssh-keygen -t ed25519 -C "your_email@example.com"**
  - Cette opération crée une clé SSH, qui utilise l'e-mail fourni comme étiquette : > Generating public/private ALGORITHM key pair.
- vous êtes maintenant invité à entrez un fichier où enregistrer la clé. On peut directement appuyer sur entré et ca va créer un fichier à l'emplacement indiqué dans le terminal ((/home/YOU/.ssh/id_ed25519) mais je conseille de créer la clée dans un dossier personnalisé au cas où vous voulez créer d'autre clé SSH. par exemeple tapez à la suite **/home/YOU/.ssh/maPremiereCle** en mettant votre nom dans la racine à la place de YOU.
- Une deuxième invitation vous est donné pour créer une phrase secrete de clé ssh. Vous pouvez directement appuyer sur entré si vous voulez pas en mettre, celan ne posera pas de problème pour la suite . Cette phrase secrete sera utilisé comme sécurité supplémentaire lorsque vous vous autenthifierai sur git pour acceder à gitHub. Encore une fois, je recommande de mettre une phrase secrète comme cle par exemple. UNe fois la clé rentré il vous est demandé de rerentrer la phrase secrete.
- Votre clé est maintenant créer vous pouvez vous en assurant en allant dans le dossier **/home/YOU/.ssh/** où vous trouverez un fichier maPremiereCle qui contient votre clé et un fichier maPremiereCle.pub qui contient la clé necessaire à GitHub pour vous reconnaitre.

Ajout de votre clé SSH à ssh-agent

- Tapez la commande
  - $ **eval "$(ssh-agent -s)"**
  - cette commande démarrez l'agent SSH en arrière-plan à qui vous pourrez confié votre clé.
  - Elle affichera en sorti l'identifiant de l'agent : Agent pid 59566 par exemple
- Ajoutez votre clé privée SSH à ssh-agent. (le '~' est un tild tapez altgr + touche 2 accent aigue)
  - $ **ssh-add ~/.ssh/id_ed25519**
  - Si vous avez créé votre clé avec un autre nom ou si vous ajoutez une clé existante portant un autre nom, remplacez id_ed25519 dans la commande par le nom de votre fichier de clé privée,(par exemple **~/.ssh/maPremiereCle**)

Ajout de la clé SSH à votre compte sur GitHub

encore une fois tout est bien fait sur la doc github ajout clé ssh github

- copier toute la sortie de la commande **cat ~/.ssh/id_ed25519.pub** (ou bien ~/.ssh/maPremiereCle.pub)
- Aller dans les paramètre de votre compte sur Github puis dans Clés SSH et GPG puis cliquer sur ajouter une nouvelle clé SSH
- Dans le champ « Titre », ajoutez une étiquette descriptive pour la nouvelle clé. Par exemple, si vous utilisez un ordinateur portable personnel, vous pouvez nommer cette clé « Ordinateur portable personnel ».
- selectionnez le type de clé : pour nous ce sera authentification
- Collez votre clé dans le champ « Clé » et ajoutez la clé en cliquant sur le bouton vert ajouter clé, vous serz ensuite invité à rentrer votre mdp gitHub.

2 Configuration de Git et ajout d'un premier repertoire GitHub sur votre ordi perso en local

- Dans un terminal configurer le fichier config de git en tapant les commandes :
  - $ **git config —global user.name yourname** (double tiret devant global)
  - $ **git config —global user.email youremail** (double tiret devant global) email de GitHub
- Vous pouvez maintenant cloner n'importe quelle repertoire public ou privé via l'accée SSH : cliquer sur l'adresse SSH du repertoire dans <>Code puis SSH puis coper le lien
- Dans un terminal tapez git clone puis le lien copié **git clone git@github.com:bob/unnomderepertoire.git**
- Si vous avez mit une phrase secrete elle vous sera demandé à cette instant précis. Si toute les étape on était correctement executé vous devriez avoir le nom du repertoire en tapant ls, vous y avez entierement accés en local.

3 Mettre à jour son dossier local par rapport au dépot GitHub

Par défaut, git renome le répertoire que vous avez cloné en origin. C'est entre autre la racine du dossier '.git' (vous pouvez le voir dans le repertoire en affichant les dossiers caché ou en tapant ls -a dans votre terminal (l'option -a est pour affiché tous ce qui se trouve dans le dossier sans exception)). Vous pouvez vérifier ce nom en tapant la commande

- $ **git remote** (affiches les nom des sources) ou **git remote -v** (affiche plus de détails sur les remotes notament le lien internet)
- $ **git status**
  - **l'une des commandes les plus importantes**
  - Permet de savoir sur quelle branche on se trouve

- Si votre répertoire vient d'être crée ou bien si vous n'avez pas encore crée de branche vous devriez vous trouver sur la branche main qui est par défaut la branche pincipale du dépot.
- Permet de savoir si le dossier local est à jour, en avance par rapport à la source ou en retard (après avoir fait git fetch, voir suite)

Rechercher les mise à jour du dépot GitHub depuis Git

Pour mettre à jour votre dossier github vous devez tout d'abord aller les chercher dans Github

- $ **git fetch origin**
  - cette commande ira chercher toutes les mise à jour de la source origin.
  - Si vous voulez seulement chercher les mise à jour d'une certaine branche par exemple de Branche1 tapez $ **git fetch origin Branche1**
  - Si vous disposez d'une autre source remote2 par exemple et que vous voulez seulement mettre à jour le main de remote2 tapez $ **git fetch remote2 main**

Suite à cette manipulation, git aura possession de toutes les mises à jours mais ne les affichera pas. Pour avoir des informations sur les mises à jour, faites un coup de **git status.** Si des modifications on été fait sur la branche à la source, vous devriez voir comme message branche en retard de x commit

Mettre à jour le dépot local

- $ **git pull origin main**
  - Permet de mettre à jour la branche main depuis la source origin
  - vous pouvez simplement tapez **git pull** pour effectuer toute les mises à jours détecter par git fetch

4 Envoyer des modifications sur GitHub depuis le dossier local