Github

- □□ Guide Complet : Utiliser Git & GitHub via SSH

? Guide Complet : Utiliser Git & GitHub via SSH

Ce guide explique comment relier une machine Linux (comme un poste de développeur, un serveur ou un Bastion) à un dépôt GitHub en utilisant une clé SSH, et détaille les commandes quotidiennes pour travailler sur son code (Clone, Pull, Commit, Push).

---

1. ? Étape préliminaire : Déclarer sa machine sur GitHub

Pour pouvoir interagir avec GitHub sans jamais avoir à taper de mot de passe, GitHub doit connaître la **clé publique** de votre machine.

A. Récupérer la clé publique locale

Sur le terminal de la machine qui va cloner le code (ex: votre Bastion), affichez votre clé publique :

```
cat ~/.ssh/id_ed25519.pub
```

Note : Si la commande vous dit que le fichier n'existe pas, générez d'abord une clé avec `ssh-keygen -t ed25519`.

Sélectionnez et **copiez l'intégralité du résultat** (qui commence par `ssh-ed25519...`).

B. Ajouter la clé sur GitHub

1. Connectez-vous à votre compte sur GitHub.
2. Cliquez sur votre photo de profil en haut à droite, puis sur **Settings**.
3. Dans la barre latérale gauche, cliquez sur **SSH and GPG keys**.
4. Cliquez sur le bouton vert **New SSH key**.
5. **Title** : Donnez un nom clair pour identifier la machine (ex: `Bastion Proxmox` ou `PC Portable Perso`).
6. **Key type** : Laissez sur *Authentication Key*.
7. **Key** : Collez votre clé publique.
8. Cliquez sur **Add SSH key**.

---

2. ? Cloner le dépôt (Le rapatriement initial)

Maintenant que GitHub connaît votre machine, vous pouvez télécharger votre code. On appelle ça un **Clone**. C'est une action que l'on ne fait qu'une seule fois par machine.

```
# Remplacer par l'URL SSH de votre propre dépôt
git clone git@github.com:VotrePseudo/nom-du-repo.git
```

Astuce : Placez-vous toujours dans le bon dossier (ex: `cd ~` ou `cd /opt`) avant de lancer cette commande, car Git créera un sous-dossier portant le nom du dépôt à cet endroit.

---

3. ?? Configuration de l'identité Git (À faire une fois)

Avant de pouvoir enregistrer des modifications (commit), Git a besoin de savoir "qui" vous êtes pour signer votre travail. Placez-vous dans votre nouveau dossier et lancez ces deux commandes :

```
cd nom-du-repo
git config --global user.name "Votre Nom ou Pseudo"
git config --global user.email "votre-email@github.com"
```

---

4. ? Le Workflow Quotidien (Le cycle de vie du code)

A. Télécharger les dernières mises à jour (PULL)

Avant de commencer à travailler, il faut toujours s'assurer d'avoir la dernière version du code (au cas où vous l'auriez modifié depuis un autre PC ou via l'interface web).

```
git pull
```

B. Vérifier l'état de ses fichiers (STATUS)

Pour voir quels fichiers vous avez modifiés, créés ou supprimés sur votre machine :

```
git status
```

C. Ajouter ses modifications (ADD)

Pour dire à Git de prendre en compte vos modifications en vue de la prochaine sauvegarde (on place les fichiers dans le "panier") :

```
# Pour ajouter un fichier précis :
git add chemin/vers/le/fichier.yml

# Pour TOUT ajouter d'un coup (le plus courant) :
git add .
```

D. Valider la sauvegarde (COMMIT)

C'est l'acte de sauvegarder le contenu de votre "panier" dans l'historique local, avec un message expliquant ce que vous avez fait.

```
git commit -m "Ajout du serveur DHCP dans l'inventaire Ansible"
```

E. Envoyer sur GitHub (PUSH)

Vos modifications sont sauvegardées sur votre PC, mais pas encore sur le serveur GitHub ! Il faut les "pousser" vers le nuage.

```
git push
```

□□ Résumé de la boucle de travail standard :
Je modifie mes fichiers ➔ `git add .` ➔ `git commit -m "Message"` ➔ `git push`.