SSH Setup machine – Version pour Windows

Objectifs

1. Vérifier et installer les prérequis.
2. Générer des clés SSH sécurisées (type **Ed25519**).
3. Configurer Git pour l'utilisation des clés SSH et la signature des commits.

Prérequis Windows

Outils nécessaires

Outil	Comment l'obtenir
Git pour Windows	https://git-scm.com/download/win
OpenSSH Client	Inclus dans Windows 10/11
Git Bash	Installé avec Git pour Windows

Qu'est-ce qu'une clé SSH dans Git ?

Une clé SSH est un identifiant d'accès pour le protocole réseau SSH (Secure Shell). Ce protocole réseau sécurisé authentifié et chiffré est utilisé pour la communication à distance entre des machines sur un réseau ouvert non sécurisé. Le protocole SSH est utilisé pour le transfert de fichiers à distance, la gestion du réseau et l'accès à distance au système d'exploitation. L'acronyme SSH est également utilisé pour décrire un ensemble d'outils permettant d'interagir avec le protocole SSH.

SSH utilise une paire de clés pour établir une liaison sécurisée entre des parties distantes. La paire de clés contient une clé publique et une clé privée. La nomenclature privée et publique peut prêter à confusion, car on parle de clés dans les deux cas. Il est plus utile de considérer la clé publique comme un « verrou » et la clé privée comme une vraie « clé ». Vous donnez le « verrou » public aux parties distantes pour chiffrer ou « verrouiller » les données. Ces données sont ensuite ouvertes avec la clé « privée » que vous détenez en lieu sûr.

Les clés SSH sont générées via un algorithme cryptographique à clé publique, le plus courant *Ed25519* de part sa petite taille (bits) et de son haut niveau de sécurité.

D'autre algorithme existe, tel que RSA très lente et potentiellement vulnérable aux attaques quantiques ou DSA plus rapide que RSA d'un niveau de sécurité modéré à élevé, mais plus lente que Ed25519 lors de l'encryption du contenu.

Vérifiez vos installations

Dans **Git Bash**, exécutez :

```
git --version
ssh -V
```

- Si Git s'exécute → OK
- Si SSH renvoie une version OpenSSH → OK

---

1. Générer une clé SSH Ed25519 sous Windows

Dans **Git Bash**, exécutez :

```
ssh-keygen -t ed25519 -C "votre_email@eduvaud.ch"
```

Appuyez sur **Entrée** pour accepter les chemins par défaut.

Vos clés seront créées ici :

```
/c/Users/<votre_nom>/.ssh/id_ed25519
/c/Users/<votre_nom>/.ssh/id_ed25519.pub
```

> *Git Bash utilise /c/Users… comme équivalent de C:\Users\…*

> A l'ETML, ton `$HOME` directory est mappé sur `H:`, donc les clés seront par défaut sur `/h/.ssh` cela dépend de comment Microsoft OneDrive, votre (répertoire réseau) a été configuré. Ce n'est pas un problème.

2. Activer et utiliser l'agent SSH (ssh-agent)

`ssh-agent` c'est quoi ?

`ssh-agent` sert à gérer et stocker vos clés privées SSH en mémoire, de manière sécurisée, pour éviter que vous deviez retaper votre passphrase à chaque utilisation.

Ajouter la clé au ssh-agent

Toujours dans **Git Bash** :

```
eval `ssh-agent`
ssh-add ~/.ssh/id_ed25519
```

> Noter que `~/` pointe automatiquement part défaut vers `/c/Users/<votre_nom>` sous Git Bash Windows. `/h` à l'ETML.

Démarrer l'agent SSH au démarrage de Windows

> **Cette partie est facultative.** Elle permet de donner les instructions à votre machine sous Windows de démarrer l'agent SSH automatiquement.

Ouvrir **PowerShell en mode Administrateur**, puis :

```
Set-Service ssh-agent -StartupType Automatic
Start-Service ssh-agent
```

Vérifier l'état :

```
Get-Service ssh-agent
```

3. Copier la clé publique

Méthode 1 — Git Bash

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

> `clip` copie dans le presse-papiers Windows.

Méthode 2 — Explorateur Windows

1. Ouvrir l'explorateur :
   `C:\Users\<vous>\.ssh\id_ed25519.pub`
2. Ouvrir avec Notepad ou autre éditeur de text tel que VS Code
3. Copier le contenu

4. Ajouter la clé publique sur GitHub

1. Sur GitHub → en haut à droite → **Settings**
2. Dans la colonne "Access" → **SSH and GPG keys**
3. Cliquer **New SSH key**
4. Title → ex. *Windows EDUVAUD*
5. Key type → **Authentication**
6. Coller la clé
7. Valider → **Add SSH key**

5. Tester la connexion SSH avec GitHub

Dans Git Bash :

```
ssh -T git@github.com
```

Première connexion → GitHub vous demande confirmation :

```
Are you sure you want to continue connecting (yes/no)?
```

Tapez : **yes**

Si tout est correct :

```
Hi USERNAME! You've successfully authenticated, but GitHub does not
provide shell access.
```

6. Cloner un repo GitHub avec SSH

Vous pouvez dès à présent cloner vos repos avec la commande :

```
git clone git@github.com:<votre-identifiant>/<votre-repo>.git
```

au lieu est place de la méthode **HTTPS**.

Bonus: Ajouter une clé publique à un serveur Linux

Les avantages d'utiliser une clé SSH pour se connecter à un serveur Linux sont :

- **Sécurité beaucoup plus élevée**
  Les clés SSH sont presque impossibles à deviner ou brute-forcer, contrairement à un mot de passe.
- **Aucune transmission de mot de passe**
  Comme il n'y a pas de mot de passe envoyé sur le réseau, le risque d'interception est quasi nul.
- **Confort d'utilisation**
  Avec un `ssh-agent`, vous n'entrez votre passphrase qu'une seule fois = connexions rapides et automatiques.
- **Protection renforcée avec passphrase**
  Même si la clé privée est volée, elle reste inutilisable sans la passphrase.
- **Automatisation et scripts**
  Les clés SSH permettent des connexions non interactives pour :
  - déploiement
  - sauvegardes
  - automatisation DevOps

1. Copier la clé publique dans le presse-papier Windows

Dans **Git Bash** :

```
clip < ~/.ssh/id_ed25519.pub
```

2. Envoyer la clé sur le serveur Linux

Depuis **Git Bash** (Windows), exécuter :

```
ssh-copy-id user@IP_du_serveur
```

> Cette commande va ajouter votre clé public sur le serveur distant dans le fichier :

```
~/.ssh/authorized_keys
```

> Noter que `~/` sous macOS/Linux correspond au répertoire `$HOME` de l'utilisateur, ici `user`, ex: `/home/user`.

3. Tester la connexion SSH sans mot de passe

Depuis Windows :

```
ssh user@IP_du_serveur
```

Si tout est correct, le login sans mot de passe fonctionne et vous vous trouvez sur la machine distante Linux.