# SSH – Référence rapide

## Installation et configuration de base SSH

**>> sudo apt install openssh-server**

L'installation comporte une **étape de génération des clefs de chiffrement**.  
Par la suite, le service SSH est lancé.

## Établissement d'une connexion SSH

Les étapes de l'établissement d'une connexion SSH sont …

1. Le **serveur** **envoie sa clef publique au client**.   
   Celui-ci **vérifie qu'il s'agit bien de la clef du serveur**, s'il l'a déjà reçue lors d'une connexion précédente ;
2. Le **client génère une clef secrète** (chiffrement symétrique) et l'envoie au serveur, en chiffrant cet échange avec la clef publique du serveur (chiffrement asymétrique).   
   Le **serveur déchiffre cette clef secrète en utilisant sa clé privée**, ce qui confirme qu'il est bien le vrai serveur ;
3. Afin de le prouver au client, il **chiffre un message standard avec la clef secrète et l'envoie au client**.  
   Si le client **retrouve le message standard en utilisant la clef secrète**, il a la preuve que le **serveur est bien le vrai serveur** ;
4. Une fois la **clef secrète échangée**, le **client et le serveur peuvent alors établir un canal sécurisé** grâce à la clef secrète commune (chiffrement symétrique) ;
5. Une fois que le **canal sécurisé est en place**, le client va pouvoir envoyer au serveur une demande d’ouverture de session et le mot de passe de l'utilisateur pour vérification.   
   Le **canal sécurisé reste en place jusqu'à ce que l'utilisateur se déconnecte**.

## Première connexion avec SSH Authentification par mot de passe

L’**authentification** **par mot de passe** est la **méthode la plus simple**.

Depuis l’hôte client …  
**>> ssh <Identifiant>@<Nom DNS ou Adresse IP du serveur>  
>> ssh tux@srv01.profsavard.info**

Si le **même identifiant est utilisé** sur le client et sur le serveur …  
**>> ssh <Nom DNS ou Adresse IP du serveur>  
>> ssh srv01.profsavard.info**

Si on doit utiliser un port différent …  
**>> ssh <Nom DNS ou Adresse IP du serveur> -p <Port>  
>>** **ssh tux@srv01.profsavard.info -p 12345**L'option -p <Port>, qui précise le port utilisé par le serveur, est facultative.   
Si rien n'est précisé, c'est le **port TCP 22** qui sera utilisé par défaut.

Pour se connecter avec SSH en IPV6 depuis un terminal ...  
**>> ssh -6 <Adresse IPv6 du serveur>  
>>** **ssh -6 tux@2a01:e35:2431::2e57**

**Remarque** ...  
Afin de pouvoir se connecter en IPV6, il faut que le serveur SSH puisse écouter les adresses IPV6.

Pour ce faire, il faut ajouter le code suivant dans le **fichier /etc/ssh/sshd\_config** sur le serveur ...  
**ListenAddress ::**

## Configuration du serveur SSH

Le **fichier de configuration du serveur SSH est /etc/ssh/sshd\_config**.

Le **répertoire /etc/ssh** est **l'emplacement de stockage des fichiers de configuration** Secure Shell.   
Le **fichier de configuration /etc/ssh/sshd\_conf** définit les options du serveur au format …

**Directive Valeur**

**Attention** ...  
Si on a modifié le fichier de configuration du serveur, il faut indiquer au service de relire son fichier de configuration …  
**>> sudo systemctl reload|restart sshd.service**

Voici les **directives les plus importantes** de ce fichier de configuration …

**Port**Cette directive indique que le **serveur SSH écoute sur le port 22**, qui est le port par défaut de SSH.   
**Port 2096**

**ListenAddress**Dans certains cas, on peut avoir plusieurs interfaces réseau (ou interfaces virtuelles) et on aimerait limiter le serveur SSH afin qu'il n'écoute que certaines des cartes réseau.   
**ListenAddress 192.168.1.100:192.168.1.101**

Il est possible d’envisager de **ne pas autoriser l'utilisateur root à se connecter** directement à l’aide du protocole SSH, en particulier si le **système est connecté à Internet**. **Autoriser la connexion root** à l’aide du protocole SSH **offre aux pirates potentiels** la

**Directive PermitRootLogin**

possibilité d’utiliser une **attaque par force brute** en essayant plusieurs mots de passe pour obtenir un accès.

Cette directive, qui est **définie par défaut avec la valeur prohibit-password**, interdit toutes les méthodes d'authentification interactives, **n'autorisant que celle utilisant une clé publique**.   
Les valeurs peuvent être **yes**, **prohibit-password**, **forced-commands-only**, ou **no**.

* Si cette option est **définie sur prohibit-password** (ou son alias obsolète without-password), le **mot de passe et l'authentification** **interactive au clavier sont désactivés** pour l’utilisateur root.
* Si cette option est définie sur **forced-commands-only,** une connexion root avec **authentification par clé publique est autorisée**, mais uniquement **si l'option de commande a été spécifiée** (ce qui peut être utile pour effectuer des sauvegardes à distance même si la connexion root n'est normalement pas autorisée). Toutes les **autres méthodes d'authentification sont désactivées** pour l’utilisateur root.
* Si cette option est définie sur **no**, root n'est pas autorisé à se connecter.

**Directive AllowUsers et DenyUsers**

Il est aussi possible de limiter seulement certains utilisateurs pouvant se connecter à l’aide de SSH  
Dans ce cas, il existe deux choix … **AllowUsers** ou **DenyUsers**.

Ces deux mots clés vous permettent de spécifier une liste de noms d'utilisateurs …  
**AllowUsers tux kermit squiddly**

* Si on utilise le **directive AllowUsers**, seuls ces utilisateurs seront autorisés à se connecter au serveur SSH ;
* Si on utilise le **directive DenyUsers**, tous les utilisateurs peuvent se connecter au serveur SSH, à l'exception des utilisateurs répertoriés.

Il est possible d’utiliser des caractères génériques pour faire correspondre les modèles …  
**DenyUsers invit\* # Refuser à tout utilisateur portant un nom commençant par « invit »  
DenyUsers ????app # Refuser à tout utilisateur avec un nom qui contient 4 caractères et qui se termine par « app »**

**Remarque** …  
Il existe également une **directive AllowGroups** et une **directive DenyGroups**.

Le tableau suivant répertorie les **diectives les plus courantes** que l’on devra peut-être modifier sur le serveur openSSH.

|  |  |
| --- | --- |
| Directive | Description |
| Banner | Spécifie le fichier à afficher avant la connexion d’un usager |
| AllowUsers | Autorise la liste d'utilisateurs spécifiée à se connecter à l'aide de SSH |
| DenyUsers | Empêche la liste d'utilisateurs spécifiés de se connecter à l'aide de SSH |
| PasswordAuthentication | Autorise l'authentification à l'aide de mots de passe textuels |
| MaxAuthTries | Spécifie le nombre de tentatives qu’un utilisateur peut entrer un mot de passe (défaut 6) |
| PermitRootLogin | Autorise le compte utilisateur root à se connecter à l'aide de SSH |
| PermitEmptyPassword | Spécifie si un usager peut se connecter sans mot de passe (défaut No) |
| PrintMotd | Spécifie si le contenu du fichier /etc/motd est affiché lorsqu’un usager se branche avec SSH |
| Protocol | Spécifie le niveau de protocole de chiffrement – Le niveau 2 est préféré et plus sécurisé |
| PubkeyAuthentication | Autorise l'authentification à l'aide de certificats |
| X11Forwarding | Autorise les serveurs X distants à exécuter des applications client X à l'aide d'un tunneling |
| AllowTcpForwarding | Autorise le serveur à accepter les protocoles en tunnel |
| listenAddress | Spécifie les interfaces qu’écoute SSH |
| Port | Modifie le serveur SSH pour écouter un autre port |

Après avoir **apporté des modifications à la configuration**, il faudra **redémarrer le serveur openSSH** pour que celles-ci prennent effet.   
Il est également possible de **simplement recharger le fichier de configuration**, cela permettra de ne pas déconnecter les utilisateurs utilisant le service.

Pour ce faire, on utilise …  
**>> sudo systemctl reload|restart sshd.service**

## Utilisation d’openSSH

Il est intéressant de connaître plusieurs commandes de client SSH …

* **ssh**  
  Cette commande permet de se connecter à un serveur SSH et de travailler sur le serveur à l’aide d’une ligne de commande ;
* **scp**  
  Cette commande permet de transférer des fichiers vers et/ou depuis un serveur SSH à l’aide de la ligne de commande ;
* **sftp**  
  Cette commande permet de se connecter à un serveur SSH et d'exécuter des commandes de type **ftp**.

On doit **connaître la fonction** de chacune de ces commandes ainsi que certaines **directives clés du fichier /etc/ssh/ssh\_config**.

### Fichier ssh\_config

La **directive Host** permet de **spécifier différentes règles pour différents serveurs SSH** auxquels on se connecte.   
Par exemple, les éléments suivants appliqueraient la valeur 0 de **ConnectTimeout** lors de la connexion à srv01.profsavard.com et la valeur 600 lors de la connexion à profsavad.info …

**Host srv01.profsavard.info  
 ConnectTimeout 0  
Host test.tux.org  
 ConnectTimeout 600**

**Remarque** …  
La **plupart des** **directives du fichier ssh\_config sont liées aux directives du fichier de configuration du serveur SSH**.

### Commande ssh

La **première fois que l’on essaie de se connecter** à un hôte à l’aide de la commande ssh, on est **invité à vérifier l'empreinte de la clé RSA** …  
**>> ssh** [**tux@srv01.monserveur.info**](mailto:tux@srv01.monserveur.info)

### Connexion à un serveur SSH

Depuis l’hôte client …  
**>> ssh <Identifiant>@<Nom DNS ou Adresse IP du serveur>  
>> ssh tux@srv01.profsavard.info  
>> ssh tux@192.168.1.100**

Si le **même identifiant est utilisé** sur le client et sur le serveur …  
**>> ssh <Nom DNS ou Adresse IP du serveur>  
>> ssh srv01.profsavard.info**   
**>> ssh 192.168.1.100**

Si on doit se **connecter sous un autre compte utilisateur**, …  
**>> ssh -l <Nom d’utilisateur>@<Nom DNS ou Adresse IP du serveur>  
>> ssh -l kermit@srv01.profsavard.info**   
**>> ssh -l kermit@192.168.1.100**

Pour exécuter une commande sur le système distant, mais revenir immédiatement à son système client, on utilise la syntaxe …  
**>> ssh <Nom DNS ou Adresse IP du serveur> <Commande>  
>> ssh 192.168.1.100 "sudo apt update && sudo apt upgrade -y"**    
**>> ssh 192.168.1.100 "sudo systemctl status sshd.service"**

### Commandes scp et sftp

Pour copier un fichier de l’hôte actuel vers un hôte distant, on utilise la syntaxe suivante …  
**>> scp <Nom du fichier> <Nom DNS ou Adresse IP du serveur>:<Répertoire>  
>> scp /etc/hosts 192.168.1.22:/tmp**  
**>> scp monfichier.texte tux@192.168.1.77:/home/tux/Documents/monfichier.texte**

Pour transférer un fichier depuis l’hôte distant vers le système local, on utilise …  
**>> scp tux@192.168.1.77:/home/tux/Documents/monfichier.texte monfichier.texte**

Une fonctionnalité pratique de scp est la possibilité de copier des fichiers entre deux serveurs distants …  
**>> scp** tux**@192.168.1.100:monfichier.texte tux@192.168.1.101:monfichier.texte**

La commande sftp se connecte à un serveur SSH et fournit une interface client de type ftp …  
**>> sftp** [**tux@srv01.profsavard.info**](mailto:tux@srv01.profsavard.info) **sftp> pwd  
sftp> ls  
sftp> get tux.texte .  
sftp> ls  
sftp> bye  
>>**

### Connexion sans mot de passe

**Étape 1**Sur l’hôte client SSH, on exécute la commande ssh-keygen.   
**>> ssh-keygen -t rsa**

Pour confirmer la présence des clés …  
**>> ls ./.ssh/**

Pour afficher le contenu de la clé publique …  
**>> cat ./.ssh/id\_rsa.pub**

**Étape 2**Il existe maintenant une **clé publique dans le fichier ~/.ssh/id\_rsa.pub**.

Celui-ci **doit être copié sur le serveur distant dans le fichier ~/.ssh/registered\_keys** de son **compte sur l’hôte distant**.  
On devra peut-être créer ce répertoire et définir les autorisations sur 770) …  
**>> ssh-copy-id -i id\_rsa.pub srv01.profsavard.info**

La clé publique va se retrouver dans le **fichier /home/lsavard/.ssh/authorized\_keys** sur le serveur.

**Étape 3**On définit les **autorisations du fichier allowed\_keys sur 640** …  
**>> ssh -l tux srv01.profsavard.info 'chmod 640 /home/tux/.ssh/authorized\_keys'**

Il est **maintenant possible de se connecter au serveur SSH sans mot de passe** …  
**>> ssh -l tux srv01.profsavard.info 'date'**

### Utilisation de la clef de chiffrement sans mot de passe

**Étape 1**on **démarre un nouveau shell bash avec l'utilitaire ssh-agent** …  
**>> ssh-agent /bin/bash**

**Étape 2**On **exécute la commande suivante** …  
**>> ssh-add ~/.ssh/id\_rsa**

À partir de ce moment, lorsque l’on **utilise le shell ssh-agent** pour se connecter à distance au serveur SSH, la **connexion a lieu sans mot de passe**.