|  |  |
| --- | --- |
| SSH Client, SSH Terminal, SSH Android, SSH Client APK SSH Download for  Android – Download SSH Client, SSH Terminal, SSH Android, SSH Client APK  Latest Version - APKFab.com | Secure Shell |

Présentation

Secure Shell (SSH) est un protocole réseau cryptographique permettant d'exploiter des services réseau en toute sécurité sur un réseau non sécurisé.   
Les applications typiques incluent la ligne de commande à distance, la connexion et l'exécution de commandes à distance.   
Néanmoins, tout service réseau peut être sécurisé avec SSH.

SSH fournit un canal sécurisé sur un réseau non sécurisé en utilisant une architecture client-serveur, connectant une application cliente SSH à un serveur SSH.   
La spécification du protocole distingue deux versions principales, appelées SSH-1 et SSH-2.   
Le port TCP standard pour SSH est le 22.   
SSH est généralement utilisé pour accéder aux systèmes d'exploitation Linux, mais il peut également être utilisé sur Microsoft Windows   
Windows 10 utilise OpenSSH comme client SSH et serveur SSH par défaut.

Génération et utilisation de paires de clés

Les mots de passe sont excellents, mais sont aussi limités.  
Beaucoup de gens utilisent des mots de passe faibles.

Avec l'accès par mot de passe activé, n'importe qui peut se connecter à un serveur depuis n'importe quel endroit avec suffisamment de temps et une puissance de traitement suffisante.

C'est pourquoi il est intéressant l’utilisation de clés.

Les clés SSH sont basées sur le concept de chiffrement à clé publique.   
Elles se divisent en deux parties …

* une moitié publique   
  et
* une moitié privée

La partie publique est déposée sur un serveur et la partie privée (protégée) est soit sur un hôte local, soit sur une clé USB sécurisée (celle-ci étant elle-même chiffrée et protégée par mot de passe).

Une fois que la moitié publique de l clé est déposée un serveur, il est possible de se connecter à l’aide du protocole SSH à un hôte distant en utilisant la moitié locale et privée de sa clé pour l'authentification.

Les clés SSH peuvent même offrir un certain degré de facilité, car la plupart des systèmes d'exploitation sont livrés avec une sorte de trousseau qui peut être automatiquement déverrouillé lors de la connexion de l'utilisateur et qui stocke les parties privées de ses clés en toute sécurité.

# Mise en place

Pour ce tutoriel, on utilise les adresses IP suivantes …

* Serveur distant … 192.168.1.100
* Client local … 192.168.1.200

**Depuis le serveur** …

On vérifie que l'adresse IP du serveur est correctement configurée à l'aide de la commande ip a …  
**>> ip a  
…  
2: ens33: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc pfifo\_fast state UP group default qlen 1000  
 link/ether 00:0c:29:d7:88:09 brd ff:ff:ff:ff:ff:ff  
 altname enp2s1  
 inet 192.168.1.100/24 brd 192.168.1.255 scope global dynamic ens33  
 valid\_lft 79758sec preferred\_lft 79758sec  
 inet6 fe80::20c:29ff:fed7:8809/64 scope link  
 valid\_lft forever preferred\_lft forever**

Il est également possible d’utiliser la commande hostname -I pour obtenir l'adresse IP d'un hôte …  
**>> hostname -I  
192.168.1.100**

**Depuis le client SSH** …

On peut vérifier, à l’aide de la commande ping si le serveur est accessible …  
**>> ping -c 4 192.168.1.100  
PING 192.180.1.100 (192.180.1.180) 56(84) bytes of data.  
64 bytes from 192.180.1.100: icmp\_seq=1 ttl=52 time=75.6 ms  
64 bytes from 192.180.1.100: icmp\_seq=2 ttl=52 time=53.4 ms  
64 bytes from 192.180.1.100: icmp\_seq=3 ttl=52 time=56.3 ms  
64 bytes from 192.180.1.100: icmp\_seq=4 ttl=52 time=51.5 ms**

**--- 192.180.1.180 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3005ms  
rtt min/avg/max/mdev = 51.458/59.186/75.612/9.637 ms**

**Depuis le client SSH** …

La partie qui suit va générer deux types de paires de clés :   
une paire de clés RAS et une paire de clés Ed25519.

## Création de clés RSA (Rivest-Shamir-Adleman)

**Depuis le client SSH**  …

Les étapes pour générer une paire de clés RSA sont …

* Génération des clés ;
* Confirmation de l'emplacement par défaut oὺ seront sauvegardées les clés ;
* Définition d’une phrase secrète.

**>> ssh-keygen -b 4096 -C "Exemple de clés RSA"  
ssh-keygen -b 4096 -C "Exemple de clés RSA"  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/tux/.ssh/id\_rsa):  
Created directory '/home/tux/.ssh'.  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/tux/.ssh/id\_rsa  
Your public key has been saved in /home/tux/.ssh/id\_rsa.pub  
The key fingerprint is:  
SHA256:ytlPyiV0w/dNWdx2yYkLCj8jS5CyYwGCIhBJe7zYJHw Exemple de clés RSA  
The key's randomart image is:  
+---[RSA 4096]----+  
|B= |  
|B + . o.+|  
|o+ E o . . . +\*|  
| B = . o.. . ..+|  
| . \* oS=+ .. ..|  
| . ...=o.oo . o |  
| +.o o . .|  
| . \* |  
| o . |  
+----[SHA256]-----+**

L'image aléatoire dans le code précédent est principalement destinée aux humains afin que les clés puissent être validées à vue.

Pour vérifier la création des clés …  
**>> ls -l .ssh  
total 16  
-rw------- 1 tux tux 746 16 déc 08:48 authorized\_keys  
-rw------- 1 tux tus 3389 16 déc 08:24 id\_rsa  
-rw-r--r-- 1 tux tux 746 16 déc 08:24 id\_rsa.pub  
-rw-r--r-- 1 tux tux 222 16 déc 08:48 known\_hosts**

La prochaine étape est de copier cette clé RSA nouvellement générée sur le serveur.

**Remarque** …  
Si on a saisi un mot de passe lors de la création des clés, on sera invité de le saisir lors de son utilisation.

**>> ssh-copy-id 192.168.1.100  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/tux/.ssh/id\_rsa.pub"  
The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.  
ECDSA key fingerprint is SHA256: SHA256:ytlPyiV0w/dNWdx2yYkLCj8jS5CyYwGCIhBJe7zYJHw.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
tux@192.168.1.100's password: <Mot de passe>**

**Number of key(s) added: 1  
Now try logging into the machine, with: "ssh '192.168.1.100'"  
and check to make sure that only the key(s) you wanted were added.**

**Empreinte numérique**

Lors de la première connexion à un hôte distant, ce dernier envoie avec sa clé publique.  
Par analogie, on imagine un échange de secrets d'affaires.   
Le conseiller indique que l’on n'a jamais rencontré cette personne auparavant et qu'il peut s'agir d'un imposteur.   
Cette alerte est toujours affichée lors d’une première connexion.  
Pour les prochains rendez-vous avec cette personne, le conseiller ne n’affichera plus ce message.

C'est ce que signifie le message.   
La personne est le serveur distant et le conseiller est le client ssh.

Si on s’est déjà connecté à ce serveur depuis cette installation de ssh, soit le serveur a été reconfiguré avec une nouvelle clé, soit quelqu'un usurpe l'identité du serveur.   
En raison de la gravité d'une attaque de l'homme du milieu, ce message met en garde contre cette possibilité.

Pour valider que l’on est connecté au bon hôte distant …  
**>> ssh-keyscan -t rsa 192.168.1.100 | ssh-keygen -lv -f -  
# 192.168.1.100:22 SSH-2.0-OpenSSH\_8.4p1 Debian-5  
3072 SHA256:n4NfQv95bX95M1lYR/qs+Qn6c8seX2gO2s4VxFwWYdw 192.168.1.180 (RSA)  
+---[RSA 3072]----+  
| .+\*|  
| o.+E|  
| +o |  
| .. o|  
| S . .=.|  
| + o .o+|  
| . = +.++\*|  
| . B.B=B@|  
| oo=.BO@|  
+----[SHA256]-----+**

Il est aussi possible de demander à l’administrateur système de l’hôte distant.  
On peut obtenir l’empreinte numérique d’un hôte à l’aide de la commande …  
**>> ssh-keygen -lv  
Enter file in which the key is (/home/tux/.ssh/id\_rsa):  
4096 SHA256:WmHWOYJz5vJclM567ingMGKOOWZS1Z2w0tZ2cmV7kkw Exemple de clés RSA (RSA)  
+---[RSA 4096]----+  
| |  
| .. . oE |  
| oo=B.\*= o |  
| o =B\*\*o.= . |  
| . o..S++ o |  
| + o .\* o |  
| \* . +..+ . |  
|=o. . .o . |  
|+. o+ |  
+----[SHA256]-----+**

Comme dernière étape, on vérifie que l’on peut accéder à l’hôte distant grâce à la clé que l’on vient de copier.

**Remarque** …  
Si on a saisi un mot de passe lors de la création des clés, on sera invité de le saisir lors de son utilisation.

**>> ssh 192.168.1.100  
Linux debian11 5.10.0-9-amd64 #1 SMP Debian 5.10.70-1 (2021-09-30) x86\_64**

**The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.**

**Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.  
Last login: Thu Dec 16 08:17:28 2021 from 192.168.1.200**

## Création de clés Ed25519 (Courbes elliptiques)

Tout comme RSA, on débute par générer une nouvelle clé, en spécifiant cette fois le type ed25519.

Les clés ed25519 sont basées sur une courbe elliptique et de certain administrateurs système pensent qu'elles offrent une sécurité supérieure à RSA.

**Remarque** …  
On ne peut pas utiliser la moitié publique d'une clé ed25519 pour chiffrer des fichiers, comme on peut le faire avec une moitié publique RSA. Il y a donc dans ce cas un compromis mais cela dépend de ses besoins.

**Depuis le client SSH** …

Pour générer une paire de clé ed25519 …  
**>> ssh**-**keygen -t ed25519 -C "Exemple de clés ed25519"  
Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/lsavard/.ssh/id\_ed25519):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/lsavard/.ssh/id\_ed25519  
Your public key has been saved in /home/lsavard/.ssh/id\_ed25519.pub  
The key fingerprint is:  
SHA256:WHhtb+GcKqjVXOXwPjrIEX9anJoXUzOg6D1PaLRM/As Exemple de clés ed25519  
The key's randomart image is:**

**+--[ED25519 256]--+  
| |  
| . . . |  
| . oo+.o. |  
| +o.=O o+ |  
| ..SB.=Oo o |  
| +o.E+X |  
| o.++.@o+ |  
| o o.=.+. |  
| . .o |  
+----[SHA256]-----+**

La prochaine étape est de copier la nouvelle clé sur l’hôte distant …  
**>> ssh-copy-id -i .ssh/id\_ed25519.pub 192.168.1.100  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: ".ssh/id\_ed25519.pub"  
…  
Number of key(s) added: 1  
…**

Une fois installé, on peut essayer de se connecter en SSH à l’hôte distant, en spécifiant la moitié la clé privée ed25519clé …  
**>> ssh 192.168.1.100 -i .ssh/id\_ed25519  
Linux debian11 5.10.0-9-amd64 #1 SMP Debian 5.10.70-1 (2021-09-30) x86\_64  
The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.**

**Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.  
Last login: Thu Dec 16 08:55:11 2021 from 192.168.1.200**

# Fonctionnement

Le principe des clés asymétriques et du chiffrement à clé publique peut être difficile à comprendre.   
Pour la plupart, on n'aura pas à se soucier des mathématiques de la génération de clés – on doit simplement savoir que l’on se retrouve toujours avec deux clés, une publique et une privée.

**Fichiers de clés publiques et privées**

On a créé une paire de clés (deux fichiers) … une moitié peut être librement transmise (la moitié publique) et l'autre que l’on garde en sécurité ailleurs (la moitié privée).

Par défaut, ces fichiers se trouvent dans le répertoire personnel (/home) de l’utilisateur, dans le répertoire .ssh (répertoire caché) …   
**>> ls -la .ssh  
total 32  
drwx------ 2 lsavard lsavard 4096 16 déc 09:11 .  
drwxr-xr-x 3 lsavard lsavard 4096 16 déc 08:18 ..  
-rw------- 1 lsavard lsavard 958 16 déc 09:20 authorized\_keys  
-rw------- 1 lsavard lsavard 419 16 déc 09:05 id\_ed25519  
-rw-r--r-- 1 lsavard lsavard 106 16 déc 09:05 id\_ed25519.pub  
-rw------- 1 lsavard lsavard 3389 16 déc 08:24 id\_rsa  
-rw-r--r-- 1 lsavard lsavard 746 16 déc 08:24 id\_rsa.pub  
-rw-r--r-- 1 lsavard lsavard 222 16 déc 08:48 known\_hosts**

Les moitiés publiques des clés se terminent par .pub, et les moitiés privées n'ont pas de suffixe de fichier.

On peut afficher le contenu de ces fichiers …  
**>> less.ssh/id\_rsa  
-----BEGIN OPENSSH PRIVATE KEY-----  
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAACFwAAAAdzc2gtcn  
NhAAAAAwEAAQAAAgEAqlio4X2lyPPczhapy/iyeFe0aVkEzz9GHluIT4nnTvkg4ooRAf+F  
9tzrb4ndPwIy/jK9ctMwQyTnn86jKCCuOYOv93i0iedSjNKT/eQy6nlVPcA65H4qf1prgn  
kPG5oKnscFA3m2bpmlTnyozZZYz+c9/SCcGQ+XW6smBlkoOAwNS8qmrtxgZzbLg2zjENO1  
GXIYxseQhWyWRq0KxxSmtj77p1Bo1m+wQ8PfoIY8aHrx2Qhu6wINfDwWc8qGdUbImritnV  
/Dgno7b5K3due32UTEOnO7LsmAknT28OihOM547XBdGAbP04jo1Onvfm6uwjidPcDLJBBj  
…  
Qh16RJ96fPyGCyY0ShVi8KzzPxq+oLV8lAjrdkDD55OUVhEsQXnpho6qbkpxDLt7o4KVNK  
qnW1xumov9mtAAAAFEV4ZW1wbGUgZGUgY2zDqXMgUlNBAQIDBAUG  
-----END OPENSSH PRIVATE KEY-----**

La moitié privée de la clé RSA est un fichier sensible. Ce fichier possède une taille de 4 096 bits.

La moitié publique de la clé RSA, qui est le fichier qui a été déposé sur l’hôte distant …  
**>> less.ssh/id\_rsa.pub  
ssh-rsa  Exemple de clés RSA**

**Remarques** …  
On retrouve également un commentaire à la fin … Exemple de clés RSA … commentaire qui a été spécifié au moment de la génération.   
De la même manière, il est possible d’afficher les clés ed25519 …  
**>> less .ssh/id\_ed\_25519   
>> less .ssh/id\_ed28819.pub**Le fichier contenant les clés ed25519 sont beaucoup plus court que son homologue RSA. Les clés ed25519 ont une longueur fixe (cela signifie également que le -b drapeau est ignoré au moment de la génération).

## Modification de la clé publique

Si pour des raisons qui sont hors de notre contrôle, l’identifiant de l’hôte distant a été modifié, on peut recevoir ce message …  
**@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @  
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!  
Someone could be eavesdropping on you right now (man-in-the-middle attack)!  
It is also possible that a host key has just been changed.  
The fingerprint for the ECDSA key sent by the remote host is SHA256:vdJTJW4ewGtOAdQXCXJ+cbjvrNm9787/CQQnCeM9fjc.  
Please contact your system administrator.  
Add correct host key in /home/vagrant/.ssh/known\_hosts to get rid of this message.  
Offending ECDSA key in /home/vagrant/.ssh/known\_hosts:1  
ECDSA host key for 192.168.33.11 has changed and you have requested strict checking.  
Host key verification failed.**

Par ce message SSH indique qu'il connaît déjà l'adresse IP de l’hôte du client SSH.   
Toutefois, le client SSH consulte son fichier known\_hosts et constate qu’elle ne correspond pas.

**Depuis l’hôte client** …

Pour remédier à cette situation, on doit effacer l'ancienne clé du fichier known\_hosts …  
**>> ssh-keygen -R <Adresse IP de l’hôte distant>  
>> ssh-keygen -R 192.168.1.100**

Lors de la prochaine connexion, il faudra accepter la nouvelle clé de l’hôte distant.

**Fichier authorised\_keys**

Lors de la connexion à un hôte distant, SSH valide l'ID de clé que l’on fournit par rapport à une liste de fichiers authorized\_keys.

En copiant sa clé publique sur l’hôte distant à l’aide de la commande ssh-copy-id, elle se retrouve dans un fichier spécifique de l'utilisateur personnel auquel on se connecte.

**Depuis le serveur** …

On retrouve ce fichier dans le répertoire personnel (/home) de l'utilisateur, dans son répertoire .ssh …  
**>> pwd  
/home/tux**

**>> ls -la .ssh/  
drwx------ 2 lsavard lsavard 4096 16 déc 09:11 .  
drwxr-xr-x 3 lsavard lsavard 4096 16 déc 08:18 ..  
-rw------- 1 lsavard lsavard 958 16 déc 09:20 authorized\_keys**

**>> less .ssh/authorized\_keys**   
**ssh-rsa  Exemple de clés RSA**

**ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGUlFKjMrUcWQeJpj09evo7DK/9gQL2gnpbQpO0na7gd Example de clés ed25519**

La première clé (RSA) est la suivante …   
**>> cat .ssh/authorized\_keys | head -n1**

La seconde (ed25519) est la suivante …  
**>> cat .ssh/authorized\_keys | tail -n1**

**Remarque** …  
il est aussi possible de copier manuellement les clés publiques dans le fichier authorized\_keys de l'hôte auquel on se connecte.   
Afin de supprimer une clé résidant sur l’hôte distant …  
On accède au serveur ;  
On accède, à l’aide d’éditeur (vim ou nano), au fichier ~/.ssh/authorized\_keys ;  
On supprime la ligne contenant la clé ;  
On sauvegarde et on quitte.

**Protection des clés privées**

SSH est sensible aux autorisations dont disposent ses fichiers.  
Il est important que sa clé privée ne soit lisible par un utilisateur aléatoire qui pourrait ouvrir une session sur le système.

En règle générale, ce ne sera pas un problème si on vient de générer ses clés, mais si on les déplace plus tard d'un hôte à un autre, il se peut que les autorisations puissent être modifiées.

**Depuis le client SSH** …

Une bonne règle de base consiste à vérifier les paramètres …  
**>> ls -lha .ssh/  
total 32K  
drwx------ 2 lsavard lsavard 4,0K 16 déc 09:11 .  
drwxr-xr-x 3 lsavard lsavard 4,0K 16 déc 08:18 ..  
-rw------- 1 lsavard lsavard 958 16 déc 09:20 authorized\_keys  
-rw------- 1 lsavard lsavard 419 16 déc 09:05 id\_ed25519  
-rw-r--r-- 1 lsavard lsavard 106 16 déc 09:05 id\_ed25519.pub  
-rw------- 1 lsavard lsavard 3,4K 16 déc 08:24 id\_rsa  
-rw-r--r-- 1 lsavard lsavard 746 16 déc 08:24 id\_rsa.pub  
-rw-r--r-- 1 lsavard lsavard 222 16 déc 08:48 known\_hosts**

Dans la commande précédente, on peut constater que les moitiés publique et privée des clés (clés id\_rsa et clés id\_ed25519) ont des valeurs différentes.

* Les moitiés publiques des clés ( \*.pub) possèdent la valeur 644 (lecture/écriture, lecture, lecture) …  
  **-rw-r--r--**
* Les moitiés privées des clés possèdent la valeur 600 (lecture/écriture, aucune, aucune) …  
  **-rw-------**

**Mot de passe ou non**

Bien que l’on puisse générer une clé sans mot de passe et qu'il existe des cas d'utilisation valides pour le faire (par exemple, dans le cas de déploiements automatisés), il est recommandé de générer sa propre clé avec un mot de passe.

Cela signifie que si une clé est protégée avec un mot de passe dans son trousseau, vous serez invité à saisir la phrase secrète pour déverrouiller la clé.   
Ceci pourrait s’avérer utile si une personne malveillante essaie d’utiliser cette clé afin d’accéder à sesdonnées.

Si on égare ou on se fait dérober sa clé privée, il faut rapidement révoquer la moitié publique.

# Commutateurs ssh

Lors de la génération des clés, on peut ajouter des commutateurs à la commande ssh-keygen.

La page de manuel de la commande peut fournir une quantité de détails pratiques …  
**>> man ssh-keygen**

Voici quelques commutateurs intéressants …

* **-b <Nombre de bits>**Ce commutateur spécifie le nombre de bits lors de la génération d’une clé (RSA ou ed25519) ; avons le drapeau de commentaire ;
* **-C <Commentaire>**  
  Ce commutateur est utilisé afin d’ajouter une description à une clé.   
  Cela peut s’avérer utile si on utilise différentes clés pour différents hôtes distants …  
  une clé Git, une clé pour un serveur personnel, une clé du serveur d'entreprise, ... ;
* **-f <Nom de fichier>**  
  Ce commutateur permet de spécifier le nom de la nouvelle clé dans la commande de génération (au lieu de le saisir lorsque l’on n’y est invité) ;
* **-l ou –lv**  
  Ce commutateur permet d’afficher l'empreinte d'une clé.   
  Cela peut être très utile pour vérifier les paires de clés ;
* **-p**   
  Ce commutateur permet de modifier la phrase secrète d'une moitié privée, sans générer une nouvelle clé ;
* **-t dsa | ecdsa | ed25519 | rsa**  
  Ce commutateur permet de spécifier le type de clé.  
  RSA est généralement la plus compatible. ed25519 peut ne pas être possible dans certains cas ;
* **-v**  
  Ce commutateur permet d’afficher une sortie verbeuse.  
  Il peut être passé plusieurs fois, chaque v augment le niveau de débogage.

# Commandes pratiques (côté client SSH)

## Fichier /etc/hosts

Pour ajouter le serveur distant à son fichier /etc/hosts …  
**>> echo "<Adresse IP> srv01" | sudo tee --append /etc/hosts  
>> echo "192.168.1.100 srv01" | sudo tee --append /etc/hosts**ou **>> echo "192.168.1.100 srv01" | sudo tee -a /etc/hosts**

**>> ping -c 4 srv01  
PING srv01 (192.168.1.100) 56(84) bytes of data.  
64 bytes from srv01 (192.168.1.100): icmp\_seq=1 ttl=64 time=0.019 ms  
64 bytes from srv01 (192.168.1.100): icmp\_seq=2 ttl=64 time=0.046 ms  
64 bytes from srv01 (192.168.1.100): icmp\_seq=3 ttl=64 time=0.048 ms  
64 bytes from srv01 (192.168.1.100): icmp\_seq=4 ttl=64 time=0.055 ms**

**--- srv01 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3007ms  
rtt min/avg/max/mdev = 0.019/0.042/0.055/0.013 ms**

**Remarque** …  
L’adresse IP peut être de version 4 ou 6.

## SSH et autre utilisateur

Si l'utilisateur avec lequel on se connecte est différent de celui que l’on utilise localement, il est possible de spécifier manuellement le nom d'utilisateur …  
**>> ssh <Nom utilisateur>@srv01  
>> ssh squiddly@srv01**ou  
**>> ssh srv01 -l <Nom utilisateur>  
>> ssh srv01 -l squiddly**

**Remarque** …  
Pour mettre fin à une connexion SSH sur un hôte distant, on entre la commande exit ou la séquence de touches CTRL+D.

## SSH et un autre port

Si le serveur SSH auquel on se connecte écoute sur un port différent, on peut spécifier ce port.  
La valeur par défaut est le port TCP 22, mais si on a modifié le port en écoute, on peut également spécifier le nouveau port …  
**>> ssh srv01 -p <Numéro de port>  
>> ssh srv01 -p 2222**

## SSH et l’'exécution d’une commande

Il est possible d'exécuter une commande sur un hôte sans à avoir de s’u connecter …  
**>> ssh srv01 <Commande>  
>> ssh srv01 "cat /etc/hostname"  
debian11**

Ceci est particulièrement utile pour les logiciels d'automatisation ou les scripts que l’on souhaite exécuter localement mais qui interagissent avec des hôtes distants.

## Fichiers de configuration

Si on possède un système sur lequel on travaille quotidiennement, afin d’éviter de manipuler SSH à l'aide d'arguments de ligne de commande, il est possible d’en automatiser la connexion.  
Il peut être avantageux de configurer sa configuration avec des arguments typiques de manière permanente.

**Méthode 01**On peut modifier, pour tous les utilisateurs du système, le fichier /etc/ssh/ssh\_config.   
**Méthode 02**On peut modifier, pour un utilisateur spécifique, le fichier ~/.ssh/ config.

**Remarque** …  
Si le fichier ~/.ssh/ config n’existe pas, il doit être créé.  
**>> ssh touch ~/.ssh/config  
>> ssh chmod 600 ~/.ssh/config  
>> ssh vim | nano ~/.ssh/config**

Pour les deux fichiers, les directives seront les mêmes.

Voici quatre exemples de bloc de construction …

* **Host \* !srv01-C   
   IdentityFile ~/.ssh/id\_ed25519  
   Port 22**
* **Host srv01-A  
   Hostname 192.168.1.100  
   User tux**
* **Host srv01-B  
   Hostname srv01  
   IdentityFile ~/.ssh/id\_rsa  
   Port 22  
   User tux**
* **Host srv01-C  
   Hostname** **fe80::c00:27ff:ab56:ff01%%ens33  
   IdentityFile ~/.ssh/id\_rsa  
   Port 22  
   User tux**

**Remarques** …  
Le premier bloc est générique (en utilisant \*) et les autres doivent être des variantes du nom srv01.  
L'entrée hôte générique (Host \*) est une entrée globale. Les paramètres de ce bloc s'appliqueront à tous les hôtes (à l'exception de srv01-C.  
Afin d’échapper le caractère % dans l’adresse IPv6 (fe80::c00:27ff:ab56:ff01%%ens33), on ajoute un %%.

## Connexions SSH verbeuses

À fin de débogage, on peut demander un détail de ce qui se passe lors d’une connexion.  
**>> ssh -v <Nom utilisateur>@srv01  
>> ssh -v squiddly@srv01**ou  
**>> ssh srv01 -vv <Nom utilisateur>  
>> ssh srv01 -vv squiddly**  
ou  
**>> ssh srv01 -vvv <Nom utilisateur>  
>> ssh srv01 -vvv squiddly**

# Commandes pratiques (côté hôte distant)

Le **fichier /etc/ssh/sshd\_config** permet de modifier les paramètres pour un hôte distant.

Les directives répertoriées sont la plupart des paramètres que le serveur SSH (sshd) lira au démarrage du système et s'applique au démon en cours d'exécution.

## Modification du port d’écoute

Pour modifier le port par défaut sur lequel le démon SSH s'exécute …  
On décommente la directive …  
**#Port 22**

On modifie la directive pour que la port d’écoute 2222 …  
**Port 2222**

Toute modification au fichier **/etc/ssh/sshd\_config** exige de recharger le fichier de configuration ou le redémarrage du démon sshd …  
**>> sudo systemctl reload|restart sshd.service**

**Remarque** …  
Cela ne devrait pas terminer la connexion car le service sshd est conçu pour que les modifications ne provoquent pas de perte d'accès, même si ces modifications empêchent de se reconnecter.

Pour se connecter à l’hôte distant, il faut maintenant indiquer spécifiquement le port sur lequel on doit se connecter …  
**>> ssh srv01 -p 2222**

**Remarque** …  
Sur le serveur SSH, on peut vérifier si le port est ouvert …  
**>> ss -nl sport = :2222  
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process  
tcp LISTEN 0 128 0.0.0.0:2222 0.0.0.0:\*  
tcp LISTEN 0 128 [::]:2222 [::]:\***

## Modification de l'adresse d'écoute

Par défaut, SSH écoute sur toutes les adresses et interfaces d’un système …  
**#AddressFamily any  
#ListenAddress 0.0.0.0  
#ListenAddress ::**

Pour modifier et que le démon n'écoute que le protocole IPv4 et n’utilise que l’interface ens33 …  
**AddressFamily inet  
ListenAddress 192.168.1.100  
#ListenAddress ::**

Remarque …  
**ListenAddress ::** est également répertorié. :: est l'équivalent IPv6 de 0.0.0.0 pour IPv4.

Toute modification au fichier **/etc/ssh/sshd\_config** exige de recharger le fichier de configuration ou le redémarrage du démon sshd …  
**>> sudo systemctl reload|restart sshd.service**

**>> ss -nl sport = :2222  
Netid State Recv-Q Send-Q Local Address:Port Peer Address:Port Process  
tcp LISTEN 0 128 192.168.1.100:2222 0.0.0.0:\***

L'option IPv6 est disparue.

## Modification du niveau de journalisation

Il existe plusieurs niveaux auxquels SSH peut se connecter, dictés par la directive **LogLevel** …  
**# Logging  
#SyslogFacility AUTH  
SyslogFacility AUTHPRIV  
#LogLevel INFO**

Les possibilités sont **QUIET**, **FATAL**, **ERROR**, **INFO**, **VERBOSE**, **DEBUG**, **DEBUG1**, **DEBUG2**, et **DEBUG3**.

**Remarque** …  
Le manuel du démon SSH répertorie que toutes les options DEBUG violent la vie privée des utilisateurs.  
Il n'est donc pas recommandé de les utiliser.

## Interdiction de la connexion root

Certaines distributions refusent la connexion root par défaut, ce qui est, dans la plupart des cas, considéré comme une bonne idée.

Pour vérifier si root peut se connecter à l’aide de SSH …  
**#LoginGraceTime 2m  
#PermitRootLogin yes  
#StrictModes yes**

On modifie la directive …  
**PermitRootLogin no**

Toute modification au fichier /etc/ssh/sshd\_config exige de recharger le fichier de configuration ou le redémarrage du démon sshd …  
**>> sudo systemctl reload|restart sshd.service**

Cela n'empêche pas une connexion locale de l’utilisateur root.  
Il est toujours possible de se connecter à une console.

## Désactivation des mots de passe

Il est possible de forcer l'utilisation de la clé.  
il n’est plus nécessaire d'autoriser l'accès par mot de passe.  
**#PermitEmptyPasswords no  
PasswordAuthentication yes**

On remplace par no …  
**PasswordAuthentication no**

Toute modification au fichier /etc/ssh/sshd\_config exige de recharger le fichier de configuration ou le redémarrage du démon sshd …  
**>> sudo systemctl reload|restart sshd.service**

## Définition d’un message du jour (motd)

Si la directive PrintMotd est défini sur yes, les utilisateurs peuvent voir le contenu de /etc/motd lorsqu'ils se connectent.

Pour afficher le message du jour à la connexion …  
**#PermitTTY yes  
PrintMotd no  
#PrintLastLog yes**

On remplace par yes …  
**PrintMotd yes**

Toute modification au fichier /etc/ssh/sshd\_config exige de recharger le fichier de configuration ou le redémarrage du démon sshd …  
**>> sudo systemctl reload|restart sshd.service**

Un message sera désormais affiché chaque fois que l’on se connecte.

Cette fonctionnalité est généralement utilisée par les entreprises pour avertir les mauvais acteurs qui tentent d'accéder à leurs systèmes.

## Directive UseDNS

Si cette option est définie sur no (valeur par défaut), seules les adresses et non les noms d'hôtes peuvent être utilisées dans les fichiers **~/.ssh/authorized\_keys** et **/etc/ssh/sshd\_config**.  
**#UseDNS no**

Lorsqu'il est défini sur no, le démon SSH ne recherchera pas le nom d'hôte distant et vérifiera que l'adresse IP distante correspond à l'adresse IP attendue, en fonction de ce nom d'hôte.

Cela signifie que lorsque UseDNS est défini sur yes et que l’hôte à partir de lequel on se connecte n'a pas d'entrée DNS inversée, SSH essaiera de faire correspondre l'adresse IP qu'il attend avec ce qu'il voit, et échouera probablement à le faire.

En pratique, tout cela signifie que si le service DNS ne fonctionne pas, on doit attendre un peu le temps que la requête DNS expire

## Autorisation ou blocage des utilisateurs

Afin de spécifier les utilisateurs auxquels on cherche à accorder un accès SSH, on doit modifier la directive AllowUsers.

**Remarque** …  
Il est possible que cette directive soit absente ou même commenté dans le fichier sshd\_config.

On prut ajouter cette dictective à la fin du fichier …  
**AllowUsers tux**

Toute modification au fichier /etc/ssh/sshd\_config exige de recharger le fichier de configuration ou le redémarrage du démon sshd …  
**>> sudo systemctl reload|restart sshd.service**

Une fois que l’on a modifié le fichier /etc/ssh/sshd\_config, il est également possible d’ajouter plusieurs noms à cette liste, ou même remplacer cette liste blanche par une liste noire à l’aide de la direcitive DenyUsers.

De plus, il est aussi possible de travailler avec des groupes (au lieu de noms d'utilisateur individuels) avec AllowGroups et DenyGroups.

# Agents SSH

Lorsque l’on se connecte sur un hôte distant à l’aide de SSH (après avoir configuré une clé) et que l’on est invité à saisir un mot de passe, ce qui est demandé en réalité est de déchiffrer la partie clé privée de sa paire de clés publique-privée (le fichier id\_rsa par défaut).  
Ce mot de passe permet de vérifier que l’on est bien celui que l’on prétend être par rapport à l'hôte distant.

Cette demande de mot de passe peut devenir fastidieux.

C'est là qu'interviennent les agents SSH. Ils sont un endroit en mémoire où sa clé privée désormais déchiffrée est rangée, une fois que l’on a fourni le mot de pase, pour la durée de la session.

Une fois l’on a modifié sa clé privée dans l’agent, l'agent est alors chargé de présenter la clé à tous les serveurs auxquels on se connecte, sans que l’on ait à saisir à nouveau sa phrase secrète.

## Mise en place

On crée une clé asymétrique à l’aide de la commande ssh-keygen …  
**>> ssh-keygen -t ed25519 -b 2048**

**>> ls -l .ssh  
total 8  
-rw------- 1 tux tux 464 19 déc 13:56 id\_ed25519  
-rw-r--r-- 1 tux tux 98 19 déc 13:56 id\_ed25519.pub**

**>> ssh-copy-id 192.168.1.100  
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/lsavard/.ssh/id\_ed25519.pub"  
The authenticity of host '192.168.1.89 (192.168.1.89)' can't be established.  
ECDSA key fingerprint is SHA256:KT2Jir12uxyvD7SCKPYtG64v++HUHjr2ab2cqrSOjb8.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes  
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed  
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys  
lsavard@192.168.1.89's password:**

**Number of key(s) added: 1**

**Now try logging into the machine, with: "ssh '192.168.1.89'" and check to make sure that only the key(s) you wanted were added.**

**>> ssh 192.168.1.100  
Enter passphrase for key '/home/tux/.ssh/id\_ed25519': …  
Linux debian11 5.10.0-9-amd64 #1 SMP Debian 5.10.70-1 (2021-09-30) x86\_64**

**The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the individual files in /usr/share/doc/\*/copyright.**

**Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.  
Last login: Sun Dec 19 13:52:03 2021 from 192.168.1.200**

**Enter passphrase for key '/home/vagrant/.ssh/id\_ed25519':**

La tentative de connexion demande le mot de passe. Ce not de passe n’est pas celui associé à la connexion mais celui de l’utilisation de la clé.

On commence par exécuter la commande ssh-agent …  
**>> ssh-agent  
SSH\_AUTH\_SOCK=/tmp/ssh-5IaHtChPUejb/agent.791; export SSH\_AUTH\_SOCK;  
SSH\_AGENT\_PID=792; export SSH\_AGENT\_PID;  
echo Agent pid 792;**

**>> pidof ssh-agent  
792 703**

On copie les différentes variables qui ont été fournies et on les saisit comme commande …  
**>> SSH\_AUTH\_SOCK=/tmp/ssh-5IaHtChPUejb/agent.791; export SSH\_AUTH\_SOCK;  
>> SSH\_AGENT\_PID=792; export SSH\_AGENT\_PID;**

Par la suite, on exécute la commande ssh-add et saisit le mot de passe …  
**>> ssh-add  
Enter passphrase for /home/tux/.ssh/id\_ed25519:  
Identity added: /home/tux/.ssh/id\_ed25519 (tux@debian11)**

La sortie indique la commande a ajouté son identité.

On ouvre une connexion sur l’hôte distant …  
**>> ssh 192.168.1.100**La connexion s’effectue sans mot de passe.

Pour arrêter (tuer – *kill*) l’agent en cours …  
**>> ssh-agent -k**  
**unset SSH\_AUTH\_SOCK;  
unset SSH\_AGENT\_PID;   
echo Agent pid 792 killed;**

Ensuite, on exécute les commandes unset suggérées afin de supprimer les variables qui ont été définies …  
**>> unset SSH\_AUTH\_SOCK; unset SSH\_AGENT\_PID;**

**Remarques** …  
Le simple fait de se déconnecter de sa session n'arrêtera pas l'exécution du programme ssh-agent.  
On ne devait pas exécuter un agent sur un hôte distant partagé entre plusieurs personnes.

Lorsque de l’exécution initial de l’agent ssh-agent, l'agent lui-même est démarré en arrière-plan et on reçoit les variables d'environnement nécessaires pour SSH.   
Une fois qu'elles sont définies, l'exécution de SSH l'amènera à lire ces variables.

Si on ajoute quelques indicateurs -vv à la commande ssh …  
**>> ssh -vv 192.168.1.100  
OpenSSH\_8.4p1 Debian-5, OpenSSL 1.1.1k 25 Mar 2021  
debug1: Reading configuration data /etc/ssh/ssh\_config  
…  
debug1: Connection established.  
…  
debug2: key: /home/tux/.ssh/id\_ed25519 (0x55b11351c410), agent  
…**

Sans l'agent chargé, mais avec la clé présente, cela ressemble à ceci …  
**debug2: key: /home/vagrant/.ssh/id\_ed25519 (0x55dea5015410)**

Les variables d'environnement SSH sont également lues par ssh-add, que l’on a utilisé pour ajouter la clé à l'agent.

L'agent d'authentification doit être en cours d'exécution et la variable d'environnement SSH\_AUTH\_SOCK doit contenir le nom de son socket pour que ssh-add fonctionne.

Si on souhaite ajouter la commande de démarrage de l'agent à un script (par exemple .bashrc), vous souhaiterez peut-être évaluer automatiquement les variables d'environnement qui vous sont fournies.

Il existe deux manières principales de configurer un agent …

* La première est que l'agent lance une nouvelle sous-commande dans laquelle certaines variables d'environnement sont exportées, par exemple ssh-agent xterm &.
* La seconde est que l'agent affihce le shell nécessaire commandes qui peuvent être évaluées dans le shell appelant comme par exemple eval 'ssh-agent -s'.

En pratique, cela signifie qu'il est plus facile de démarrer l'agent comme suit …  
**>> eval $(ssh-agent)  
Agent pid 6896**