# SSH

Pour éviter de confirmer l'ajout du fingerprint au fichier known_hosts ajouter l'option :

```
ssh <user>@<serveur> -o StrictHostKeyChecking=no
```

## Configuration

### Fichier de configuration générale

#### /etc/ssh/sshd_config

```
ListenAddress 192.168.0.50  N’écoute que depuis ces interfaces.
ListenAddress 192.168.1.50
LoginGraceTime 20       Si pas de connexion avant 20s le serveur se
déconnecte
PermitRootLogin no      Interdiction de se connecter en tant que root
PasswordAuthentication no   Pas d’identification par mot de passe
UseDNS no           Pas de reverse DNS lors de la connexion
# ajouter à la fin du fichier :
AllowUsers administrateur   autoriser uniquement les utilisateurs listés
AddressFamily inet      uniquement IPV4
```

### Fichier de configuration par serveur

#### ~/.ssh/config

```
Host <nom_host>
        Hostname <adressIP>
        User <user>
        IdentityFile <chemin vers clef ssh privée au format openssh>
        Port <port>
```

Avec la configuration de ce fichier on pourra remplacer la commande :

```
ssh -i <chemin vers clef ssh privée au format openssh> <user>@<adresseIP>
```

par

```
ssh <nom_host>
```

Le fichier de configuration sera pris en compte par la commande rsync.

### Echange de clefs

#### Génération de la clef

```
ssh-keygen –t rsa [-C commentaire]
```

Le paramètre -C permet l'ajout d'un commentaire en fin de ligne pour distinguer les clefs dans le fichier authorized_keys. Par défaut il s'agit souvent d'un username@hostname.

#### Transfert de la clef

Copier la clef générée (~/.ssh/id_rsa.pub) vers la machine à laquelle on veut donner l'accès (dans le fichier ~/.ssh/authorized_keys).

Le répertoire .ssh ne doit pas être accessible par un autre utilisateur. Si le répertoire est créé manuellement penser à corriger les droits d'accès par un chmod

Possibilité d'utiliser directement la commande :

```
ssh-copy-id -i ~/.ssh/id_rsa.pub user@host
```

---

Haut de page

## Accès sftp resteint

### Création utilisateur

```
sudo groupadd pproject
sudo useradd -g pproject -d /home/pproject -m -s /bin/false pproject
sudo passwd pproject
sudo chown root:pproject /home/pproject
sudo chmod 755 /home/pproject
sudo mkdir /home/pproject/11-Projects
chown pproject:pproject /home/pproject/11-Projects
```

```
chmod 755 /home/pproject/11-Projects
```

### Configuration SSH

Dans le fichier /etc/ssh/sshd_config mettre en commentaire la ligne :

```
#Subsystem sftp /usr/lib/openssh/sftp-server
```

Ajouter à la **fin** du fichier :

```
Subsystem sftp internal-sftp
Match User pproject
  ChrootDirectory %h
  ForceCommand internal-sftp
  X11Forwarding no
  AllowTCPForwarding no
```

Tout ce qui suit la ligne Match jusqu'à la fin du fichier ou la prochaine occurrence de Match concerne cette restriction.

ChrootDirectory : le répertoire doit appartenir à root:root avec droits 755. Le contenu appartient à l'utilisateur avec droits 775.

### Dépannage

En cas de problème consulter les erreurs dans le fichier /var/log/auth.log.

---

Haut de page

## Tunnel SSH

La machine A ne peut pas accéder directement à la machine C mais accède à la machine B qui elle peut accéder à la machine C.

### Création du tunnel

Sur la machine B créer un tunnel vers C en utilisant un port pour la redirection :

```
ssh –L <Serveur B>:<port>:<Serveur C>:22 <user>@<Serveur C>
```

Possibilité d'ajouter les options suivantes :

```
-N : pour ne pas passer de commande sur la machine cible.
-f : la commande est exécuté en tâche de fond et on récupère la main.
```

> Si on lance en tâche de fond avec l'option -f il faudra killer le process pour fermer le tunnel

## Maintien du tunnel

- **ServerAliveInterval** : durée en secondes après laquelle si aucune donnée n'a été reçue du serveur, un message sera envoyé au serveur. Par défaut 0 indiquant pas d'envoi de messages. Uniquement version 2 du protocole.
- **ServerAliveCountMax** : valeur par défaut 3. Si ServerAliveInterval=15 et ServerAliveCountMax=3, si le serveur ne répond plus, il y aura déconnexion au bout de 45s. Uniquement version 2 du protocole.

## Exemples

### Redirige le port mySQL 3306 local sur le port 3306 de la machine distante

```
#! /bin/bash
while :
do
    echo "Ouverture du tunnel $(date)"
    ssh -L 3306:127.0.0.1:3306 -N -o ServerAliveInterval=5 -o
ServerAliveCountMax=2 <user>@<IP address>
    echo "Fermeture du tunnel $(date)"
    sleep 1
done
```

### Tunnel inverse

Imaginons que la machine B n'accède pas à la machine A mais la machine A accède à la machine B.

- Sur machine A : ouverture d'un tunnel depuis la machine B via le port 30022 distant sur le port 22 local

```
ssh -fN -R 30022:localhost:22 <user B>@<machine B>
```

- Sur machine B : connexion en ssh sur machine A

```
ssh -p 30022 <user A>@localhost
```

## Vérification

On pourra vérifier l'ouverture du port par la commande netstat ou ss :

```
netstat -tlna
ss -tlna
```

## Connexion via le tunnel

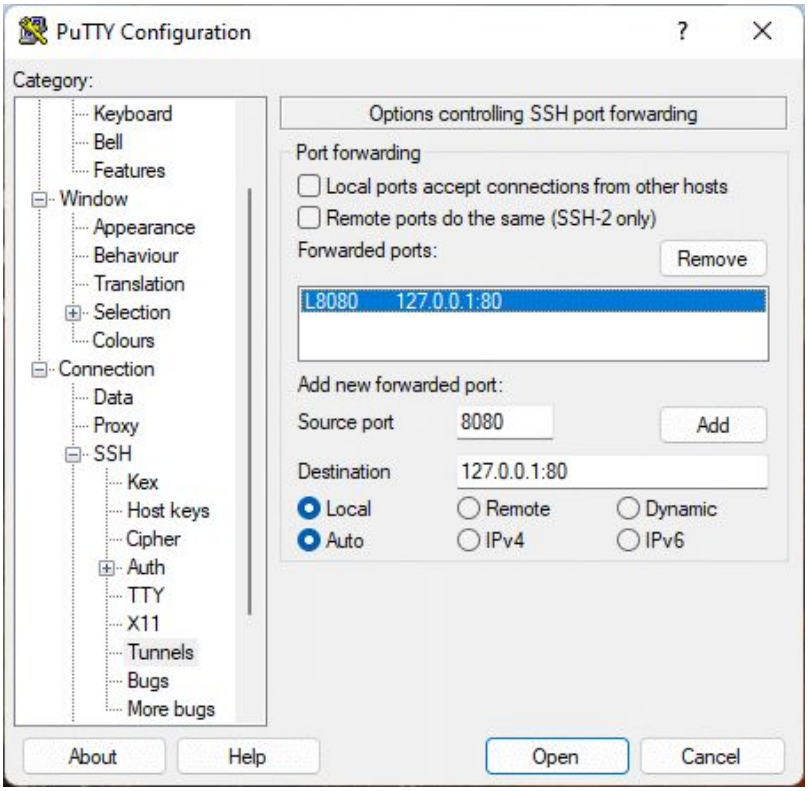
Sur la machine A accéder à C au travers de B et du port choisi :

```
ssh <Serveur B> -p <port>
```

## Tunnel ssh via Putty

Dans cet exemple on va se connecter en http (port 80) sur un serveur distant en passant outre un éventuel firewall qui bloquerait le http.

- Dans la fenêtre putty renseigner les éléments habituels (IP, port, ...)
- Aller dans la rubrique « Connection / SSH /Tunnels »
- Renseigner « Source port » avec un port qui va être utilisé localement pour le rediriger (ex 8080)
- Renseigner « Destination » avec l'adresse locale distance (exemple 127.0.0.1 ou une adresse privée accessible de la machine distante)
- Ajouter derrière cette adresse le numéro de port (par exemple :80)
- Cliquer sur le bouton « Add » puis lancer la session



- Le tunnel est ouvert il suffit de lancer un navigateur avec l'url http://localhost:8080

L'utilisation des port jusqu'à 1024 nécessite des droits administrateur

---

Haut de page

# Montage réseau au travers de SSH

## Prérequis

L'utilisateur doit appartenir au groupe fuse :

```
adduser <utilisateur> fuse
```

## Utilisation

### Montage

```
sshfs <@IP>:<chemin> <point de montage>
```

### Démontage

```
fusermount -u <point de montage>
```

---

Haut de page

# Putty à partir de Windows

## Connexion par clef

### Génération clef

Génération de clef avec puttygen.exe. Sauvegarde clef privée et clef publique. Copier/coller de la clef publique depuis la zone pour OpenSSH dans le fichier **~/.ssh/authorized_keys**

#### Utilisation de la clef

Change Settings…/Connection/Data remplir Auto-login username avec le nom d'utilisateur/ Change Settings…/Connection/SSH/Auth indiquer le chemin pour accéder au fichier de la clef privée clef.ppk

Utilisation possible en ligne de commande avec psftp pour faire du transfert de fichiers sécurisé en sftp avec nom de session mémorisé :

```
psftp <nom session>
```

### X11 forwarding

Dans Connection/SSH/X11 cocher « Enable X11 forwarding » et mettre 127.0.0.1:0.0 pour « X display location ».

Lancer X-Ming avec « Multiple windows » / « Start no client » puis « Suivant » et « Terminer ».

---

Haut de page

## Convertir format clef

### Installation outils

```
sudo apt install putty-tools
```

### Utilisation

#### Putty vers openssh

- Générer la clef privée

```
puttygen clef.ppk -O private-openssh -o id_dsa
```

- Générer la clef publique

```
puttygen clef.ppk -O public-openssh -o id_dsa.pub
```

#### openssh vers Putty

```
puttygen id_rsa -o cleprive.ppk
```

Haut de page

# Dépannage

## Debug

### Côté client

Pour obtenir plus d'information ajouter l'option verbose :

```
ssh -v <@IP>
```

### Côté serveur

Ajouter la ligne suivante dans le fichier **/etc/ssh/sshd_config**

```
LogLevel VERBOSE
```

> Les valeurs possibles sont QUIET, FATAL, ERROR, INFO, VERBOSE, DEBUG, DEBUG1, DEBUG2, and DEBUG3.

Les informations sont dans le fichier de log :

```
tail -f /var/log/secure ⇒ Redhat/CentOS
tail -f /var/log/syslog ⇒ Debian/Ubuntu
```

## L'identification par clef ne fonctionne pas

### Restriction d'accès aux clefs

Vérifier que le droits d'accès au répertoire ~/.ssh sont bien restreints au seul propriétaire. Corriger par la commande :

```
chmod 0700 ~/.ssh
```

### Restriction à la console

Vérifier l'absence de la ligne suivante dans le fichier **/etc/default/login**

```
CONSOLE=/dev/
```

sinon l'accès ne peut se faire que depuis la console système.

#### ping ne fonctionne pas

Si même le ping ne fonctionne pas, vérifier les fichiers **/etc/hosts**, **/etc/ethers** et **dhcpd.conf** qui peuvent contenir des informations contradictoires et, dans le doute, interdire la connexion par sécurité.

#### En désespoir de cause

Supprimer le répertoire ~/.ssh et faire un ssh-keygen pour laisser le système créer le répertoire avec les droits restreints.

### Lenteur de connexion

Lors de connexion le serveur ssh distant lance une résolution DNS inverse qui peut être longue (surtout si aucun serveur DNS n'est accessible).
Pour gagner du temps il est possible de désactiver cette procédure en ajoutant la ligne suivante dans le fichier **/etc/ssh/sshd_config** :

```
UseDNS no
```

Redémarrer le service ssh.

Si les symptômes persistent ajouter également la ligne suivante :

```
GSSAPIAuthentication no
```

ou bien en ligne de commande

```
ssh -o GSSAPIAuthentication=no user@serveur
```

### no matching host key type found. Their offer: ssh-dss

#### Connexion sur OS6850

Ajouter les options suivantes :

```
ssh -o HostKeyAlgorithms=+ssh-dss -m hmac-md5 <@IP>
```

#### Connexion sur OA C7000

Créer un fichier **~/.ssh/config**

```
Host 10.35.132.9
        HostkeyAlgorithms +ssh-dss
```

## Connexion aléatoire

#### Problème

La connexion ssh fonctionne de façon aléatoire en Wifi (voir pas du tout) alors que cela fonctionne en filaire. Par ailleurs les requêtes http passent sans problème en Wifi. Problème rencontré sur Raspberry Pi.

#### Désactiver la mise en veille du Wifi

Interroger l'interface sans fil

```
iwconfig
```

Vérifier la ligne Power Management (on ou off) ou interroger directement :

```
iw wlan0 get power_save
```

Désactiver la mise en veille :

```
iw wlan0 set power_save off
```

A positionner éventuellement dans le fichier /etc/rc.local.

#### Baisser la mtu sur l'interface

Essayer de baisser la mtu sur l'interface :

```
ifconfig wlan0 mtu 1200
```

A positionner éventuellement dans le fichier /etc/rc.local.

#### Qualité de service

Ajouter la ligne suivante dans le fichier /etc/ssh/sshd_config

```
IPQoS cs0 cs0
```

puis redémarrer le service :

```
systemctl restart sshd
```

## Déconnexion

En cas de déconnexion trop rapide il est possible de customiser le délai.

### sshd_config

Dans le fichier /etc/ssh/sshd_config modifier les 2 variables suivantes :

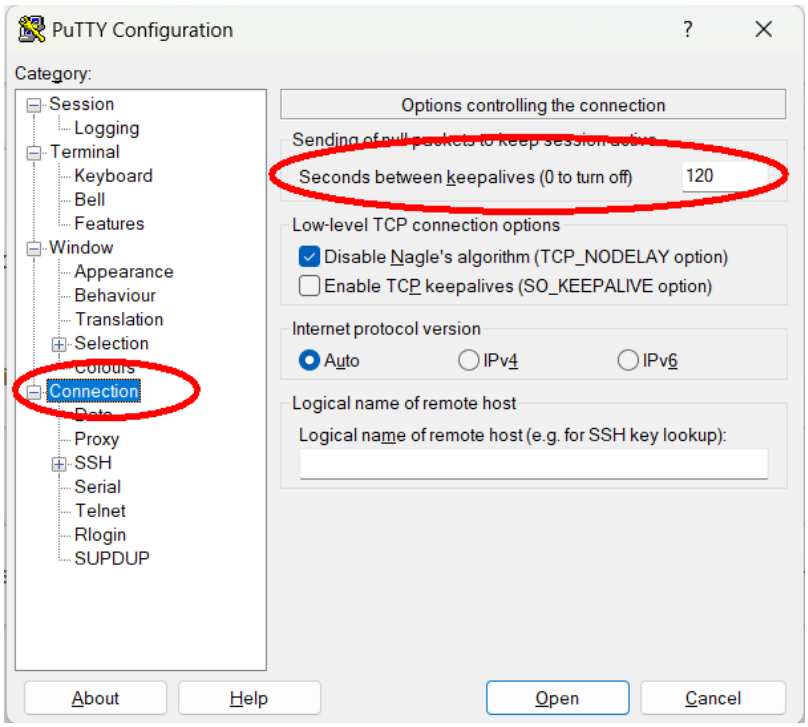
```
ClientAliveInterval 1200
ClientAliveCountMax 3
```

La valeur du délai d'expiration sera de 1200 secondes * 3 soit 1 h. On peut obtenir le même résultat en spécifiant le paramètre ClientAliveInterval seul :

```
ClientAliveInterval 3600
#ClientAliveCountMax 3
```

### Putty

Avec Putty il est également possible de spécifier le paramètre « Seconds beetween keepalives (0 to to turn off) » dans la catégorie « Connection » pour maintenir la session ouverte.



## Pseudo-terminal will not be allocated because stdin is not a terminal

Solution : essayer ssh avec l'option -t

---

Haut de page