| S T NSI | Réseau | ***TP N° SSH*** |
|---|---|---|

*Objectifs :*

SSH (Secure SHell) permet de se connecter de façon sécurisée à un système Unix, Linux et Windows.
Il faut distinguer :

- **SSH** : le protocole de communication
- **ssh** : le programme client permettant de se connecter au serveur
- **sshd** : le serveur (ssh daemon) écoutant sur le port 22 par défaut

**sshd** est un serveur qui permet de communiquer de façon sécurisée, en établissant un canal de communication entre lui et ses clients

- Il fonctionne seul et écoute les requêtes adressées par les clients au port 22

# 1 Connexion en SSH sous linux :

## 1.1 Démarrer le service ssh :

Pour démarrer le service ssh :

```
pi@raspberrypi:~ $ sudo service ssh start
```

Pour vérifier si le service est actif :

```
Sudo service ssh status
```

```
pi@raspberrypi:~ $ sudo service ssh status
● ssh.service - OpenBSD Secure Shell server
   Loaded: loaded (/lib/systemd/system/ssh.service; disabled; vendor preset: ena
   Active: active (running) since Wed 2020-12-02 16:19:38 CET; 12s ago
     Docs: man:sshd(8)
           man:sshd_config(5)
  Process: 918 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
 Main PID: 919 (sshd)
    Tasks: 1 (limit: 3861)
   CGroup: /system.slice/ssh.service
           └─919 /usr/sbin/sshd -D

déc. 02 16:19:38 raspberrypi systemd[1]: Starting OpenBSD Secure Shell server...
déc. 02 16:19:38 raspberrypi sshd[919]: Server listening on 0.0.0.0 port 22.
déc. 02 16:19:38 raspberrypi sshd[919]: Server listening on :: port 22.
déc. 02 16:19:38 raspberrypi systemd[1]: Started OpenBSD Secure Shell server.
```

## 1.2 Récupérer votre adresse IP

Bien entendu, votre Raspberry Pi doit être connectée à internet mais que vous soyez connecté en wi-fi ou via un câble ethernet n'a pas vraiment d'importance.
Pour obtenir votre adresse ip, ouvrez un terminal et tapez

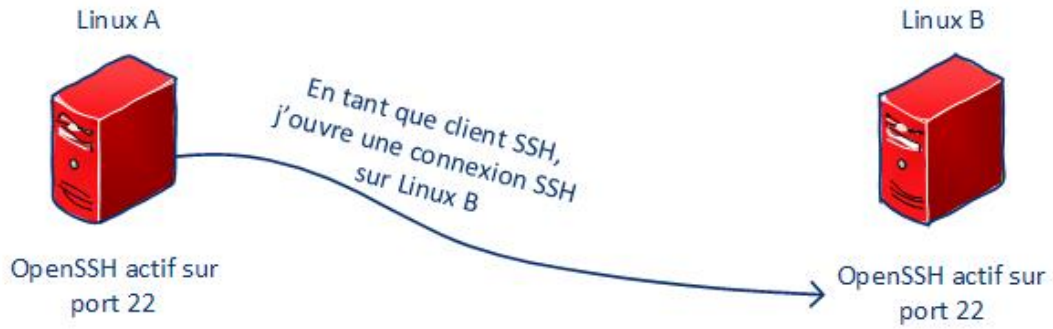
## ifconfig

```
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 169.254.119.99  netmask 255.255.0.0  broadcast 169.254.255.255
        inet6 fe80::2297:6c26:f4e5:20d3  prefixlen 64  scopeid 0x20<link>
        ether dc:a6:32:24:ce:73  txqueuelen 1000  (Ethernet)
        RX packets 1012  bytes 110195 (107.6 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 179  bytes 17064 (16.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Boucle locale)
        RX packets 68  bytes 6460 (6.3 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 68  bytes 6460 (6.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether dc:a6:32:24:ce:74  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

### 1.3 Connexion client-serveur

Il faut ici bien comprendre un principe dans la notion de "Client - Serveur".
Dans le cadre de SSH, un client peut devenir serveur et inverse.

Si nous disposons de deux machines Linux, chacune peut avoir un service OpenSSH actif et en écoute. Ces deux machines sont alors des serveurs SSH. Néanmoins, je peux décider, avec ma machine Linux A, d'aller ouvrir une connexion SSH vers ma machine Linux B. Linux A est alors client, Linux B est alors serveur :




Si, à l'inverse, depuis Linux B, je décide d'ouvrir une connexion SSH vers ma machine Linux A, c'est ma Linux B qui est le client SSH, Linux A est alors serveur.

Passons à l'action, sur une machine Linux avec "openssh-client" d'installé, on va saisir la commande suivante :

`ssh utilisateur@adresse_ip_ou_url_serveur`

Bien sûr, il faut que "utilisateur" soit remplacé par un utilisateur valide et que "monserveur" soit remplacé par le nom ou l'IP d'un serveur Linux ayant un OpenSSH à l'écoute. Si vous ne disposez que d'une machine et que celle-ci à justement un serveur OpenSSH à l'écoute, on peut initier une connexion SSH vers son propre serveur, cela à titre d'exemple seulement car ça ne présente aucun intérêt en situation opérationnelle :

`ssh utilisateur@adresseIP`

```
ssh pi@169.254.189.76
The authenticity of host '169.254.189.76 (169.254.189.76)' can't be established.
ECDSA key fingerprint is SHA256:RvXLLd+kaDquy0fS6QnkdaDH0p+c6/Rmgm35LmbsTqE.
Are you sure you want to continue connecting (yes/no)? yes
Please type 'yes' or 'no': yes
Warning: Permanently added '169.254.189.76' (ECDSA) to the list of known hosts.
pi@169.254.189.76's password:
raspberry
Warning: Permanently added '169.254.189.76' (ECDSA) to the list of known hosts.

Linux raspberrypi 5.4.79-v7l+ #1373 SMP Mon Nov 23 13:27:40 GMT 2020 armv7l

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: Wed Dec  2 16:17:11 2020

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new
password.
```

**Entrer les commandes suivantes :**

`apt-cache show openssh-client`

```
pi@raspberrypi:~ $ apt-cache show openssh-client
Package: openssh-client
Source: openssh
Version: 1:7.9p1-10+deb10u2
Architecture: armhf
Maintainer: Debian OpenSSH Maintainers <debian-ssh@lists.debian.org>
```

```
Installed-Size: 3009
Depends: adduser (>= 3.10), dpkg (>= 1.7.0), passwd, libc6 (>= 2.28), libedit2 (>=
2.11-20080614-0), libgssapi-krb5-2 (>= 1.17), libselinux1 (>= 1.32), libssl1.1 (>=
1.1.1), zlib1g (>= 1:1.1.4)
Recommends: xauth
Suggests: keychain, libpam-ssh, monkeysphere, ssh-askpass
Conflicts: sftp
Replaces: ssh, ssh-krb5
Provides: rsh-client, ssh-client
Multi-Arch: foreign
Homepage: http://www.openssh.com/
Priority: standard
Section: net
Filename: pool/main/o/openssh/openssh-client_7.9p1-10+deb10u2_armhf.deb
Size: 670064
SHA256: ed537b03803301397a28d2f473bfd0b18685dad3c0bea5b6c2eaca8b42ba0387
SHA1: 7a5f06187d6480ce3e25a7b28663aec852ede253
MD5sum: 17395fc4501ef3e6545fd1c425400aaf
Description: secure shell (SSH) client, for secure access to remote machines
 This is the portable version of OpenSSH, a free implementation of
 the Secure Shell protocol as specified by the IETF secsh working
 group.
 .
 Ssh (Secure Shell) is a program for logging into a remote machine
 and for executing commands on a remote machine.
 It provides secure encrypted communications between two untrusted
 hosts over an insecure network. X11 connections and arbitrary TCP/IP
 ports can also be forwarded over the secure channel.
 It can be used to provide applications with a secure communication
 channel.
 .
 This package provides the ssh, scp and sftp clients, the ssh-agent
 and ssh-add programs to make public key authentication more convenient,
 and the ssh-keygen, ssh-keyscan, ssh-copy-id and ssh-argv0 utilities.
 .
 In some countries it may be illegal to use any encryption at all
 without a special permit.
 .
 ssh replaces the insecure rsh, rcp and rlogin programs, which are
 obsolete for most purposes.
Description-md5: 8cde3280ebad71c16b3e8c661dae6c6d
```

apt-cache show openssh-server

```
pi@raspberrypi:~ $ apt-cache show openssh-server
Package: openssh-server
Source: openssh
Version: 1:7.9p1-10+deb10u2
Architecture: armhf
Maintainer: Debian OpenSSH Maintainers <debian-ssh@lists.debian.org>
Installed-Size: 1307
Depends: adduser (>= 3.9), dpkg (>= 1.9.0), libpam-modules (>= 0.72-9), libpam-runtime
(>= 0.76-14), lsb-base (>= 4.1+Debian3), openssh-client (= 1:7.9p1-10+deb10u2),
openssh-sftp-server, procps, ucf (>= 0.28), debconf (>= 0.5) | debconf-2.0, libaudit1
(>= 1:2.2.1), libc6 (>= 2.28), libcom-err2 (>= 1.43.9), libgssapi-krb5-2 (>= 1.17),
libkrb5-3 (>= 1.13~alpha1+dfsg), libpam0g (>= 0.99.7.1), libselinux1 (>= 1.32),
libssl1.1 (>= 1.1.1), libsystemd0, libwrap0 (>= 7.6-4~), zlib1g (>= 1:1.1.4)
Recommends: default-logind | logind | libpam-systemd, ncurses-term, xauth
Suggests: molly-guard, monkeysphere, rssh, ssh-askpass, ufw
```

```
Conflicts: sftp, ssh-socks, ssh2
Replaces: openssh-client (<< 1:7.9p1-8), ssh, ssh-krb5
Provides: ssh-server
Multi-Arch: foreign
Homepage: http://www.openssh.com/
Priority: optional
Section: net
Filename: pool/main/o/openssh/openssh-server_7.9p1-10+deb10u2_armhf.deb
Size: 290932
SHA256: 96e022a6f4d1e57c7278dfe641113e82b0ebd1aedc02d395f4779992e0beaf9b
SHA1: 6821b4c8c040f2e51976bc8f79218d87a0ece5e8
MD5sum: 73a8edc14235e3d1830eea475c0eb00b
Description: secure shell (SSH) server, for secure access from remote machines
 This is the portable version of OpenSSH, a free implementation of
 the Secure Shell protocol as specified by the IETF secsh working
 group.
 .
 Ssh (Secure Shell) is a program for logging into a remote machine
 and for executing commands on a remote machine.
 It provides secure encrypted communications between two untrusted
 hosts over an insecure network. X11 connections and arbitrary TCP/IP
 ports can also be forwarded over the secure channel.
 It can be used to provide applications with a secure communication
 channel.
 .
 This package provides the sshd server.
 .
 In some countries it may be illegal to use any encryption at all
 without a special permit.
 .
 sshd replaces the insecure rshd program, which is obsolete for most
 purposes.
Description-md5: 842cc998cae371b9d8106c1696373919
```

# 2 Travail en binôme : client serveur

## 2.1 Session de travail cliente

1. Pour en savoir plus sur le serveur ssh, consulter *man ssh*. Les fichiers de configuration sont placés dans */etc/sshd*
2. Connectez-vous au compte  du serveur voisin.
3. Vous pouvez ouvrir une session root sur le serveur, avec la commande **su**, mais attention ! respectez l'environnement
4. Y passer quelques commandes (sans rien abimer !).
5. Pour terminer une session de travail ssh, taper **exit**

## 2.2 Examen du serveur

1. Votre serveur sshd est-il actuellement en exécution sur votre machine ? Comment feriez-vous pour arreter ou démarrer ce service
2. Plus généralement, pour connaitre TOUS les serveurs réseaux à l'écoute, et ceux qui sont actifs, utilisez netstat -ntl (ou netstat -tl). Plus d'infos : consulter le man(ual)

3. Utiliser **last | less** sur votre machine locale pour connaitre les clients qui se sont connectés récemment sur votre serveur sshd

### 2.3 Transfert de fichiers par scp

scp (*secure copy*), permet de copier des fichiers et des arborescences, en utilisant **ssh** pour sécuriser les transferts

1. syntaxe générale:
   **scp [-r] source destination**, où source et destination désigne l'ensemble des fichiers à copier ou le répertoire d'accueil.
   Si les fichiers sont locaux, on utilise la syntaxe habituelle.
   S'ils sont distants, la notation est celle de ssh : **user@serveur:fichiers**
2. Exemples :
   scp -r user@serveur:fichiers  rep-local , pour copier du serveur distant les fichiers vers le répertoire *rep* d'accueil local
   scp -r fichiers-locaux  user@serveur:rep , pour copier les fichiers locaux vers le répertoire situé sur le serveur distant

### 2.4 Application

On a vu comment "prendre" la main sur une console distante avec ssh, et faire exécuter à distance des applications. Voici comment faire pour faire afficher des applications sur sa console.

```
# tout le monde est autorisé à utiliser le serveur X
[user@local]$ xhosts +
[user@local]$ ssh login@serveur
[login@serveur]$ export DISPLAY=iplocal:0.0
[login@serveur]$ xclock &   # pour essayer
[login@serveur]$ konqueror &    # pour travailler
```

### 2.5 Eteindre le raspberry à distance :

```
sudo shutdown now
```

TORNIOR / GOURDIER