Rsync

Table des matières

1 Installation

Les deux paquets *openssh-client* et *openssh-server* réuinis dans le paquet *ssh*

```
# apt-get install ssh
```

2 Connexion normale avec clé

Un client *ssh* se connecte en utilisant une clé privé. Le serveur doit disposer de sa clé publique dans le fichier */.ssh/authorized_keys*

bibi@narval se connecte sur yabon@max :

- côté client :

```
$ ssh-keygen -t dsa
```

- côté serveur :

```
$ mkdir .ssh
$ touch .ssh/authorized_keys
$ chmod 644 .ssh/authorized_keys
$ ssh login@client "cat ~/.ssh/id_dsa.pub" >> ~/.ssh/authorized_keys
```

3 Connexion d'appoint avec clé

root@max se connecte sur rsync@narval : `root@max:# ssh -i .ssh/rsync_dsa rsync@narval`

- côté client :

```
# ssh-keygen -t dsa -f .ssh/rsync_dsa
```

- côté serveur :

```
$ mkdir .ssh
$ touch .ssh/authorized_keys
$ chmod 644 .ssh/authorized_keys
$ ssh root@max "cat ~/.ssh/rsync_dsa.pub" >> ~/.ssh/authorized_keys
```

Dans le cas suivant, bien qu'en apparence la clé soit généré sur le serveur, il ne s'agit que d'une défiguration du cas ci-dessus. **A éviter**.

- côté serveur :

```
$ ssh-keygen -t dsa
$ cat .ssh/id_dsa.pub > .ssh/authorized_keys
```

- côté client :

```
# rsync -e ssh -au rsync@narval:.ssh/id_dsa .ssh/rsync_dsa
```

4 Tunnel ssh

- Ici, la machine `localhost` crée un (ou plusieurs) tunnel vers une (ou plusieurs) machines distantes.

```
$ ssh -v -N [-L 2222:narval.tk:222]* nroche@localhost
                                                    $ ssh -p 2222 root@localhost
                                                    coyote #
^C
```

- fichier *rr /.ssh/config*

```
Host lyoac??
        User acqilc
#       ProxyCommand ssh roche@polui07 echo %h:%p
        ProxyCommand ssh roche@polui07 nc %h %p
        ForwardX11 yes
```

5 Proxy ssh

Ici, la machine `localhost` crée un proxy sur la machine distante accessible via le port 1081.

```
$ ssh -v -D 1081 narval.tk
```