Configurer son client SSH

Introduction

Il est possible de passer des options lorsqu'on invoque la commande `ssh`, par exemple la commande :

```
$ ssh -X -p 222 root@leconteneurduprof.netlib.re
```

indique que :

- le nom de domaine du conteneur sur lequel se connecter (`leconteneurduprof.netlib.re`)
- le nom d'utilisateur (`root`)
- le port TCP vers lequel établir une connexion (`222`, par défaut le serveur SSH écoute sur le port `22`)
- l'activation du `X-forwarding` qui permet de téléporter l'affichage des applications graphiques de la machine distante vers la machine locale (option `-X`)

Seulement, c'est un peu lourd de devoir taper tout ça à chaque fois, et on peut définir des ensembles d'options dans un fichier de configuration.

Trouver et lire la documentation

Si on recherche dans le manuel toutes les pages dont le descriptif contient la chaîne de caractères "ssh" :

```
$ man -k ssh
```

on obtient quelque chose comme :

```
authorized_keys (5)  - OpenSSH SSH daemon
git-shell (1)        - Restricted login shell for Git-only SSH access
rlogin (1)           - OpenSSH SSH client (remote login program)
rsh (1)              - OpenSSH SSH client (remote login program)
slogin (1)           - OpenSSH SSH client (remote login program)
ssh (1)              - OpenSSH SSH client (remote login program)
ssh-add (1)          - adds private key identities to the authentication agent
ssh-agent (1)        - authentication agent
ssh-argv0 (1)        - replaces the old ssh command-name as hostname handling
ssh-copy-id (1)      - use locally available keys to authorise logins on a remote machine
ssh-keygen (1)       - authentication key generation, management and conversion
ssh-keyscan (1)      - gather SSH public keys
ssh-keysign (8)      - ssh helper program for host-based authentication
ssh-pkcs11-helper (8) - ssh-agent helper program for PKCS#11 support
ssh_config (5)       - OpenSSH SSH client configuration files
sshd (8)             - OpenSSH SSH daemon
sshd_config (5)      - OpenSSH SSH daemon configuration file
```

On voit qu'il existe une page de manuel de la section `5` (qui correspond aux formats de fichiers) `ssh_config` décrivant les fichiers de configuration du cliend SSH :

```
ssh_config (5)       - OpenSSH SSH client configuration files
```

Consultons cette page de manuel :

```
$ man 5 ssh_config
```

On y apprend :

- que les users peuvent ajouter leurs données de configuration dans le fichier `~/.ssh/config`
- que les commentaires sont introduits par `#` (classique). Ça peut être utile pour décrire chaque bloc de configuration, mais aussi pour désactiver temporairement une option et de pouvoir la réactiver en enlevant le `#` qui est au début de la ligne.
- que le fichier est une succession de déclarations (une par ligne) de la forme `MorClef valeur`.

- que deux mots clefs particuliers `Host` et `Match` servent à introduire un ensemble d'options relatives à un nom d'hôte donné dans la commande `ssh` ou à un ensemble de contitions plus complexe.

  Ainsi, la déclaration `Host bibi` permet de spécifier un ensemble d'options qui seront automatiquement ajoutés lors de l'exécution d'une commande `ssh bibi`.

- suit la liste des options, avec leur descriptif et valeurs par défaut.

En pratique

Dans la pratique, on écrit une succession de blocs `Host` indépendants pour décrire les différentes connexions ssh que l'on souhaite effectuer. Ainsi, la commande utilisée dans l'introduction :

```
$ ssh -X -p 222 root@leconteneurduprof.netlib.re
```

peut être remplacée par :

```
$ ssh plop
```

dès que votre fichier `~/.ssh/config` contient le bloc :

```
Host plop
    Hostname leconteneurduprof.netlib.re
    ForwardX11 yes
    Port 222
    User root
```

Exemples issus du cours

- Si vous voulez vous connecter en tant que `root` sur votre conteneur dont vous ne connaissez que l'adresse IPv6 :

  ```
  Host conteneur
      Hostname  2001:910:1410:523::fada:1234:5678
      User root
  ```

  Vous pouvez alors lancer la commande :

  ```
  $ ssh conteneur
  ```

  qui sera équivalente à :

  ```
  $ ssh root@2001:910:1410:523::fada:1234:5678
  ```

- Si votre conteneur est identifié par le nom de domaine `exemple.netlib.re`, vous pouvez remplacer le bloc précédent par :

  ```
  Host conteneur
      Hostname exemple.netlib.re
      User root
  ```

- Si vous n'avez pas de connexion IPv6 et que vous voulez passer à travers le proxy `tor` qui écoute sur le port `9050` en utilisant la commande `nc` fournie par le paquet `netcat-openbsd` :

  ```
  Host conteneurtor
      Hostname 2001:910:1410:523::fada:1234:5678
      ProxyCommand nc -x localhost:9050 %h %p
      User root
  ```

- Si vous voulez spécifier l'utilisation de `tor` pour vos connexions vers les conteneurs de tou·tes les étudiant·es (invitations), vous pouvez ajouter un bloc avec wildcard (fr: joker, étoile) :

  ```
  Host 2001:910:1410:523::fada:*
      ProxyCommand nc -x localhost:9050 %h %p
  ```

  Ainsi, lorsque vous ferez, par exemple :

  ```
  $ ssh guest@2001:910:1410:523::fada:1234:5678
  ```

  votre client ajoutera automatiquement l'option `ProxyCommand` définie dans le bloc précédent.

- Pour vous connecter à la machine `sercalssh` de l'institut Galilée qui est accessible depuis internet et sert de passerelle pour les machines des salles de TP (qui n'ont pas d'adresses IPv4 publiques) :

  ```
  Host sercalssh
      Hostname sercalssh.ig-edu.univ-paris13.fr
      User 12345678
  ```

  Pour vous connecter sur les machines des salles de TP, il faut passer à travers la machine `sercalssh` (option `ProxyJump`), et on souhaite pouvoir téléporter l'affichage des applications graphiques comme `marionnet` (option `ForwardX11`) :

  ```
  Host f2* g2* F2* G2*
      ProxyJump sercalssh
      ForwardX11 yes
      User 12345678
  ```

  Ainsi, lorsque vous tapez :

  ```
  $ ssh F209-11
  ```

  Le client va trouver que le bloc `f2* g2* F2* G2*` correspond à ce nom d'hôte, il va donc lancer la connexion à travers `sercalssh`.

  Remarque : `ProxyJump sercalssh` est essentialement une simplification de l'option `ProxyCommand ssh sercalssh -W %h:%p`.

- La clef utilisée par défaut est `~/.ssh/id_rsa`. Si vous gérez plusieurs pairs de clefs, pour utiliser la clef `~/.ssh/autre_clef` vous pouvez utiliser l'option :

  ```
  IdentityFile ~/.ssh/autre_clef
  ```

Superposition des configurations

Le début du manuel explique aussi les priorités des options : si vous passez une option dans la ligne de commande SSH, sa valeur est prioritaire sur la valeur donnée dans votre fichier de configuration personnel `~/.ssh/config` qui est elle-même prioritaire sur la configuration au niveau du système définie dans le fichier `/etc/ssh/ssh_config`, qui est elle-même prioritaire sur la valeur par défaut.

Ainsi, si par exemple le fichier `/etc/ssh/ssh_config` contient le bloc

```
Host *
    ForwardX11 yes
```

et que votre fichier `~/.ssh/config` contient le bloc

```
Host conteneur
    Hostname exemple.netlib.re
    Port 222
    User root
```

alors, la commande

```
$ ssh guest@conteneur
```

va se connecter sur le port 222 de la machine dont le nom de domaine est `exemple.netlib.re` en tant qu'user `guest`, en activant l'affichage des applications graphiques distantes.

Au delà de la commande ssh

Ces configurations seront utilisées par les programmes qui utilisent le protocole `SSH`, comme `scp`, `rsync` ou même `git`, par exemple :

```
$ scp devoir.tar.gz guest@conteneur:/opt/
```

```
$ rsync -av conteneur:/etc/nginx/sites-available .
```

```
$ git clone gituser@conteneur:depot.git
```