Installer parallel-ssh

pssh est un outil en ligne de commande pour exécuter ssh en parallèle sur plusieurs hôtes. Ses spécialités comprennent:

- Envoi de données à tous les clients
- Saisie unique d'un mot de passe pour ssh
- Enregistrement de la sortie dans des fichiers ou visionnage direct
- Automatisation des tâches **sysadmin** telles que l'application de correctifs aux serveurs, les mises à jour, l'installation de paquets, la configuration,...
- Envoi de fichiers à tous les serveurs
- Gestion des processus
- Compatible avec tous les Linux, Unix et freeBSD

Configurer les postes clients

Tâches à réaliser sur chaque client. Ecrire un script d'automatisation afin d'alléger cette tâche.

Installer le paquet openssh-server

```
# aptitude install openssh-server
```

Chaque client possédera donc son propre serveur ssh.

configurer openssh-server

Configurer le serveur ssh pour qu'il accepte les connexions root et les autorisations par clefs et non mot de passe

```
# nano /etc/ssh/sshd_config
```

Mettre à jour les lignes suivantes avec ces entrées

```
PermitRootLogin yes
PubkeyAuthentication yes
```

Démarrer le service ssh ou le redémarrer

```
# service ssh start
# service ssh restart
```

Les clients sont prêts à l'emploi

Configuration de la machine maître

Installer pssh

```
# aptitude install pssh
```

pssh / parallel-ssh

Sous debian sid, l'utiliaire **pssh** s'appelle **parallel-ssh** (sous d'autres distributions c'est simplement **pssh**).

Ici nous utiliserons **parallel-ssh** par défaut (adapter à votre configuration)

Utiliser une paire de clef publique/privée pour l'identification ssh

Construire sa clef , sauf si on en a déjà une (compte $USER, non en root)

```
$ ssh-keygen

$ ls -l .ssh/
total 12
-rw------- 1 ragnarok cyrille 1876 juin  26  2019 id_rsa
-rw-r--r-- 1 ragnarok cyrille  397 juin  26  2019 id_rsa.pub
-rw-r--r-- 1 ragnarok cyrille 2220 mars  14 16:42 known_hosts
```

La clé est créée, on la garde sous le coude ;)

Le fichier de configuration du poste maître

Ce fichier renferme la liste des hosts vers lesquels les commandes seront envoyées.

Ici, cette liste sera stockée dans **/etc/ssh/pssh_host** mais vous pouvez créer un fichier de ce type où vous le souhaitez (souvent, le fichier **~/.pssh_hosts_files** est utilisé)

Créer un fichier contenant les hosts

```
mkdir /etc/ssh/pssh_host
nano /etc/ssh/pssh_host/pssh
###Mettre les adresses IP des serveurs à administrer ici.
root@192.168.0.11
root@192.168.0.12
```

```
root@192.168.0.23
root@192.168.0.43
...
```

Si vous souhaitez utiliser les noms des machines plutôt que leurs adresses IP, il faudra modifier et adapter le fichier **/etc/hosts**

Exporter la clef publique sur les clients

```
ssh-copy-id root@192.168.0.11
ssh-copy-id root@192.168.0.12
ssh-copy-id root@192.168.0.23
ssh-copy-id root@192.168.0.43
...
```

Autant de fois que d'entrées dans votre fichier **/etc/ssh/pssh_host/pssh**

```
$ ssh-copy-id root@192.168.0.43
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed:
"/home/ragnarok/.ssh/id_rsa.pub"
The authenticity of host '192.168.0.43 (192.168.0.43)' can't be established.
ECDSA key fingerprint is SHA256:2SC8wDSd7m7UrqCRnmz6jsY+6K9GL9zkMPQZGxEQM6k.
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
root@192.168.0.43's password:
Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@192.168.0.43'"
and check to make sure that only the key(s) you wanted were added.
```

Déployer pssh

Initialiser pssh

Utilisez **ssh-agent** pour vous authentifier automatiquement (avec un nom de shell comme argument pour que les variables d'environnement de l'agent soient définies dans ce nouveau shell). Ajoutez la clé avec **ssh-add** et tapez votre mot de passe une seule fois.

```
$ ssh-agent bash
$ ssh-add
Enter passphrase for /xxxx/.ssh/identity:
```

Utiliser parallel-ssh

```
$ parallel-ssh -i  -h /etc/ssh/pssh_host/pssh_hosts uname -a
[1] 15:21:04 [SUCCESS] root@192.168.0.11
FreeBSD asgard-freeBSD 12.0-RELEASE-p13 FreeBSD 12.0-RELEASE-p13 GENERIC
amd64
[2] 15:21:04 [SUCCESS] root@192.168.0.43
Linux Tinuviel-debianStable 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2
(2019-11-11) x86_64 GNU/Linux
```

On remarque ici **l'option -i** qui permet de visualiser ce que retourne le terminal ssh des postes clients

Pour **rediriger la sortie ssh des clients vers un fichier**, on utilisera l'**option -o**

```
$ parallel-ssh -o /tmp/uname  -h /etc/ssh/pssh_host/pssh_hosts uname -a
[1] 15:22:47 [SUCCESS] root@192.168.0.11
[2] 15:22:48 [SUCCESS] root@192.168.0.43
```

Et pour visualiser les sorties.

```
$ cat /tmp/uname/root@192.168.0.11 /tmp/uname/root@192.168.0.43
FreeBSD asgard-freeBSD 12.0-RELEASE-p13 FreeBSD 12.0-RELEASE-p13 GENERIC
amd64
Linux Tinuviel-debianStable 4.19.0-6-amd64 #1 SMP Debian 4.19.67-2+deb10u2
(2019-11-11) x86_64 GNU/Linux
```

Copier des fichiers vers les clients

```
Syntaxe
```

```
$ parallel-scp -h /fichier_de_conf_pssh source destinatire
```

Exemple

```
$ parallel-scp  -h /etc/ssh/pssh_host/pssh_hosts  $HOME/test.txt /tmp/
```

Tuer des processus sur les postes clients

Syntaxe

```
$ parallel-nuke -h /fichier_de_conf_pssh nom_du_processus
```

Exemple

```
$ parallel-nuke  -h /etc/ssh/pssh_host/pssh_hosts  nginx
```