Fabric

Introduction

- Fabric : http://docs.fabfile.org/

C'est un outil permettant d'exécuter et de faire des opérations à distance sur plusieurs machines en utilisant ssh. C'est un canevas reposant sur ssh pour faire des opérations sur des groupes de machines rassemblées par exemple par rôles.

Utilisation personnelle

Exploration des machines à ajouter au fichier `known_hosts` de ssh

Voir la solution de la page Accepter automatiquement les certificats ssh des machines. La meilleure solution est de parcourir l'ensemble de machines en utilisant l'utilitaire d'*openssh* : *ssh-keyscan*.

Pour commencer, comme les machines auxquelles nous voulons accéder ne sont pas dans le fichier `~/.ssh/known_hosts`, il faut faire une exploration préliminaire acceptant toutes les clés de machine sans avoir à les approuver.

```
# exemple avec un fichier de hosts à la *ansible*
time grep -v '^#\|^[[:space:]]*$\|^\[[^]]*\]$' ~/ansible/hosts-def/hosts | ssh-keyscan -H -f - > ~/.ssh
/known_hosts
```

Définir les groupes de machines

Il arrive qu'un ensemble logique de machines (ayant le même rôle) soit sur un même VLAN et donc dans une plage d'adresses IP contiguës. Pour avoir une liste de machines, voici le type de recherche qui peut être faite :

```
for i in $(seq 1 32) ; do dig +short -x 192.168.1.$i ; done | sed -e 's/,$//' > ~/fabric/hosts.role
```

Ensuite, l'appel à fabric peut se faire comme suit :

```
fab --fabfile=~/fabric/fabfile.py --ssh-config-path=~/fabric/.ssh_config.exploration-hosts \
 --connection-attempts=3 --user=monlogin -i ~/.ssh/id_rsa \
 --hosts=$(for h in $(cat ~/fabric/hosts.role|grep -v '^#'); do echo -n "$h," ; done | sed -e 's/,$//') \
 --exclude-hosts=toto.mondomaine.fr -- ls -al .
```

- `--fabfile, -f` : le fichier définissant les actions possibles à faire sur les hôtes distants (optionnel, par défaut : `fabfile.py`.
- `--ssh-config-path` : le fichier de configuration du client ssh (notamment pour ne pas avoir à valider l'ajout des clés d'hôtes). Optionnel.
- `--connection-attempts, -n` : si une erreur réseau survient, le nombre de tentatives à faire. Optionnel, par défaut 1.
- `--user, -u` : l'utilisateur de connexion sur la machine distante.
- `--exclude-hosts, -x` : si on veut exclure quelques machines de la liste précédente.
- `--hosts, -H` : la liste des machines sur lesquelles doivent être exécutées les opérations (une liste de machine avec une virgule comme séparateur).
- `-i` : le fichier de la clé privée de l'utilisateur (nécessitera de rentrer la passephrase). Optionnel.

La liste des hôtes dans le fichier `hosts.roles` ressemble à quelque chose comme ça :

```
# quelques éventuels commentaires
host1.mondomaine.fr
host2.mondomaine.fr
host3.mondomaine.fr
host4.mondomaine.fr
host5.mondomaine.fr
host6.mondomaine.fr
host7.mondomaine.fr
```

Il faut transformer ça en :

```
host1.mondomaine.fr, host2.mondomaine.fr, host3.mondomaine.fr, host4.mondomaine.fr, host5.mondomaine.fr, host6.
mondomaine.fr, host7.mondomaine.fr
```

C'est le rôle de la commande suivante :

Supprime les commentaires et la dernière virgule en trop

```
for h in $(grep -v '^[[:space:]]*#\|^[[:space:]]*$' ~/fabric/hosts.role); do echo -n "$h," ; done | sed -e 's/,$
//'
```

Dans le cas de la commande précédente, ce qui est après « -- » est la commande distante qui sera exécutée en tant qu'utilisateur *monlogin*. C'est un moyen rapide et facile pour exécuter de petites commandes en vitesse. Pour quelque chose de plus évolué, le mieux est de créer des fonctions dans le fichier `fabfile.py`.

Un équivalent (en gros) de la commande précédente serait :

```
fab --fabfile=~/fabric/fabfile.py --ssh-config-path=~/fabric/.ssh_config.exploration-hosts \
 --connection-attempts=3 --user=monlogin -i ~/.ssh/id_rsa \
 --hosts=$(for h in $(cat ~/fabric/hosts.role|grep -v '^#'); do echo -n "$h," ; done | sed -e 's/,$//') \
 --exclude-hosts=toto.mondomaine.fr ls_al
```

Avec `ls_al`, la fonction définie dans `fabfile.py`. Pour avoir une liste des fonctions utilisables venant du fichier `fabfile.py`, il suffit d'utiliser l'option `--list, -l`.

fabfile.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from fabric.api import *

def ls_al():
    """Liste le contenu du répertoire."""
    run("ls -al")

def host_type():
    """Détermine le type et le numéro de version du noyau de la machine distante"""
    run("uname -s -r -m -p -o)
```

Utiliser un fichier de configuration pour fabric plutôt que des options sur la ligne de commande

Pour cela, définir un fichier `~/.fabricrc` comme suit :

~/.fabricrc

```
user=monlogin
connection_attempts=3
#ssh_config_path=~/fabric/.ssh_config.exploration-hosts

# on fait les traitement en parralèle
#parallel=True
# combien de machines en parallèle ?
#pool_size=3

# pas de terminaison en cas d'erreur
warn_only=True

fabfile=~/fabric/fabfile.py

# ne pas avoir stdout et stderr fusionnés sur stdout
#combine_stderr=False

# le fichier avec la clé privée
key_filename=~/.ssh/id_rsa

# pour utiliser un fichier de config ssh de l'utilisateur
use_ssh_config=True

# pour avoir un buffer par ligne (ne permet pas l'usage du prompt)
#linewise=True
#reject_unknown_hosts=True
#disable_known_hosts=True
```

Références

- Le site de *fabric* : http://fabfile.org/
- Le wiki de *fabric* : http://wiki.fabfile.org/

Pages utiles

- Un ensemble de recettes à utiliser pour *fabric* : http://wiki.fabfile.org/Recipes
- Les fichiers *fabfile* utilisés pour la gestion du site de *fabric* :https://github.com/fabric/fabric/tree/master/fabfile
- Quelques exemples d'utilisation avancée de *fabric* par son créateur : http://tav.espians.com/fabric-python-with-cleaner-api-and-parallel-deployment-support.html

Voir aussi

- http://ansible.github.com/ – Un outil qui semble bien plus adapté et intéressant que fabric.
- http://rexify.org/ – Un équivalent à *fabric* en perl.

En pages internes

- Ansible -- page ancienne