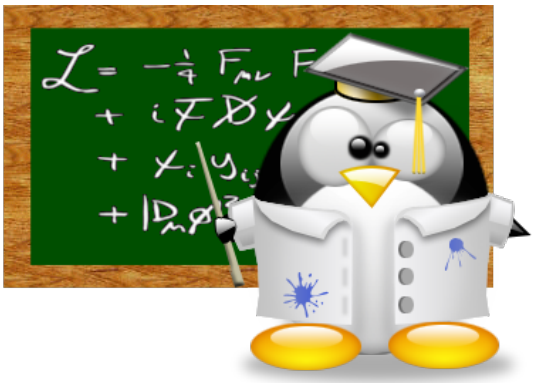

Ansible

24/06/2017

Ansible centralise et automatise les tâches d'administration. Il est sans agent (il ne nécessite pas de déploiements spécifiques sur les clients) et utilise le protocole SSH pour configurer à distance les clients Linux.

L'interface graphique web d'Ansible est payante.

L'ouverture des flux SSH vers l'ensemble des clients depuis le serveur Ansible font de lui un elément critique de l'architecture qu'il faudra attentivement surveiller.

Pourquoi scripter en Bash n'est pas considéré comme de l'automatisation ? (langage Impératif vs Déclaratif)

Même si la connaissance du Bash est une exigence de base pour un administrateur système, celui-ci est un langage de programmation interprété "**impératif**". Il exécute les instrutions les unes à la suite des autres.

Les langages dédiés au domaine (DSL Domain Specific Language) comme Ansible ou Puppet, quant à eux, ne spécifient pas les étapes à réaliser mais l'état à obtenir.

Parce qu'Ansible est un langage déclaratif, il est très simple. Il suffit de lui dire "Fais cette action" ou "Met le serveur dans cet état". Ansible considèrera l'état désirée indépendamment de l'état initial ou du contexte. Peu importe dans quel état le serveur se situe au départ, les étapes à franchir pour arriver au résultat, le serveur est mis dans l'état désiré avec un rapport de succès (avec ou sans changement d'état) ou d'échec.

La même tâche d'automatisation en bash nécessiterait de vérifier tous les états possibles, les autorisations, etc. afin de déterminer la tâche à accomplir avant de lancer l'exécution de la commande, souvent en imbriquant de nombreux "si", ce qui complique la tâche, l'écriture et la maintenance du script.

1. Installation sur le serveur

Ansible est disponible dans le dépot EPEL :

- Installation d'EPEL :

```
$ sudo yum install epel-release
```

La configuration du serveur se situe sous /etc/ansible.

Deux fichiers de configuration :

- Le fichier de configuration principal *ansible.cfg* : commandes, modules, plugins, configuration ssh ;

- Le fichier de gestion des machines clientes *hosts* : déclaration des clients, des groupes.

Le fichier /etc/ansible/hosts

```
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the 'webservers' group

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

# If you have multiple hosts following a pattern you can specify
# them like this:

## www[001:006].example.com

# Ex 3: A collection of database servers in the 'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there are no
# leading 0s:

## db-[99:101]-node.example.com
```

2. La commande ansible

La commande ansible lance une tâche sur un ou plusieurs hôtes cibles.

Syntaxe de la commande ansible

```
ansible <host-pattern> [-m module_name] [-a args] [options]
```

Exemples :

- Lister les hôtes appartenant à un groupe :

```
ansible {{group}} --list-hosts
```

- Pinger un groupe d'hôtes avec le module ping :

```
ansible {{group}} -m ping
```

- Afficher des facts d'un groupe d'hôtes avec le module setup :

```
ansible {{group}} -m setup
```

- Exécuter une commande sur un groupe d'hôtes en invoquant le module command avec des arguments :

```
ansible {{group}} -m command -a '{{commande}}'
```

- Exécuter une commande avec les privilèges d'administrateur :

```
ansible {{group}} --become --ask-become-pass -m command -a '{{commande}}'
```

- Exécuter une commande en utilisant un fichier d'inventaire personnalisé :

```
ansible {{group}} -i {{inventory_file}} -m command -a '{{commande}}'
```

Table 1. Options principales de la commande ansible

Option	Information
-a 'arguments'	Les arguments à passer au module.

Option	Information
-b -K	Demande un mot de passe et lance la commande avec des privilèges supérieurs.
--user=utilisateur	Utilise cet utilisateur pour se connecter à l'hôte cible au lieu d'utiliser l'utilisateur courant.
--become -user=utilisateur	Exécute l'opération en tant que cet utilisateur (defaut : root).
-C	Simulation. Ne fait pas de changement sur la cible mais la teste pour voir ce qui devrait être changé.
-m module	Exécute le module appelé

3. Gestion des clients

Les serveurs clients doivent être ajoutés dans le fichiers /etc/ansibles/hosts.

Un groupe "centos6" est créé :

Le fichier /etc/ansible/hosts

```
[centos6]
172.16.1.217
172.16.1.192
```

3.1. Tester avec le module ping

Par défaut la connexion par mot de passe n'est pas autorisée par Ansible.

Décommenter la ligne suivante de la section [defaults] dans le fichier de configuration /etc/ansible/ansible.cfg :

```
ask_pass      = True
```

Lancer un ping sur chacun des serveurs du groupe CentOS 6 :

```
# ansible centos6 -m ping
SSH password:
172.16.1.192 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
172.16.1.217 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Le mot de passe root des serveurs distants vous est demandé, ce qui pose un problème de sécurité...

3.2. Authentification par clef

L'authentification par mot de passe va être remplacée par une authentification par clefs privée/publique beaucoup plus sécurisée.

3.2.1. Création d'une clef SSH

La bi-clefs va être générée avec la commande **ssh-keygen** :

```
# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:RpYuJzkkaeZzve8La8Xd/8kTTE8t43DeS+L7WB26WF8 root@ansible-srv
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|     .   .       |
|    = . +       .|
|   + o *    . +.o|
|    o * S. . *o*.|
|     o * .o ..=+=|
|        o.  .oooE|
|        .+  o.*o+|
|       ...+o +o=+|
+----[SHA256]-----+
```

La clef publique peut être copiée sur les serveurs :

```
# ssh-copy-id root@172.16.1.192
# ssh-copy-id root@172.16.1.217
```

Re-commenter la ligne suivante de la section [defaults] dans le fichier de configuration /etc/ansible/ansible.cfg pour empêcher l'authentification par mot de passe :

```
#ask_pass      = True
```

3.2.2. Test d'authentifcation par clef privée

Pour le prochain test, le module shell, permettant l'exécution de commandes à distance, est utilisé :

```
# ansible centos6 -m shell -a "uptime"
172.16.1.192 | SUCCESS | rc=0 >>
 12:36:18 up 57 min,  1 user,  load average: 0.00, 0.00, 0.00

172.16.1.217 | SUCCESS | rc=0 >>
 12:37:07 up 57 min,  1 user,  load average: 0.00, 0.00, 0.00
```

Aucun mot de passe n'est demandé, l'authentification par clef privée/publique fonctionne !

3.3. Exemple de connexion à une instance Cloud Amazon ECS

Lors de la création d'une instance Amazon, une clef privée est créée et téléchargée sur le poste local.

Ajout de la clef dans l'agent SSH :

```
ssh-add path/to/fichier.pem
```

Lancement des facts sur les serveurs aws :

```
ansible aws --user=ec2-user --become -m setup
```

4. Utilisation

Ansible peut être utilisé depuis l'interpréteur de commandes ou via des playbooks.

4.1. Les modules

La liste des modules classés par catégories se trouve à l'adresse http://docs.ansible.com/ansible/modules_by_category.html. Ansible en propose plus de 750 !

Un module s'invoque avec l'option -m de la commande ansible

4.1.1. Exemples d'installation logiciel

Le module yum permet d'installer des logiciels sur les clients cibles :

```
# ansible centos6 -m yum -a name="httpd"
172.16.1.192 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
      ...
      \n\nComplete!\n"
    ]
}
172.16.1.217 | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
      ...
    \n\nComplete!\n"
    ]
}
```

Le logiciel installé étant un service, il faut maintenant le démarrer avec le module service (centos 6) ou systemd (centos 7) :

```
# ansible centos6 -m service -a "name=httpd state=started"
172.16.1.192 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
172.16.1.217 | SUCCESS => {
    "changed": true,
    "name": "httpd",
    "state": "started"
}
```

4.2. Les playbooks

Les playbooks ansible décrivent une politique à appliquer à des systèmes distants, pour forcer leur configuration. Les playbooks sont écris dans un format texte facilement compréhensible regroupant un ensemble de tâches : le format yaml.

En savoir plus sur le yaml : http://docs.ansible.com/ansible/YAMLSyntax.html

Syntaxe de la commande ansible-playbook

```
ansible-playbook <fichier.yml> ... [options]
```

Les options sont identiques à la commande ansible.

La commande renvoi les codes d'erreurs suivants :

Table 2. Codes de sortie de la commande ansible-playbook

0	**OK ou aucun hôte correspondant**
1	Erreur
2	Un ou plusieurs hôtes sont en échecs
3	Un ou plusieurs hôtes ne sont pas joignables
4	Erreur d'analyse
5	Mauvaises options ou options incomplètes
99	Execution interrompue par l'utilisateur
250	Erreur inattendue

4.2.1. Exemple de playbook apache et mysql

Le playbook suivant permet d'installer apache et mysql sur nos serveurs cibles :

```
---
- hosts: centos6
  remote_user: root

  tasks:
  - name: ensure apache is at the latest version
    yum: name=httpd,php,php-mysqli state=latest
  - name: ensure httpd is started
    service: name=httpd state=started
  - name: ensure mysql is at the latest version
    yum: name=mysql-server state=latest
  - name: ensure mysqld is started
    service: name=mysqld state=started
```

L'exécution du playbook s'effectue avec la commande **ansible-playbook** :

```
$ ansible-playbook test

PLAY [centos6] ****************************************************************

TASK [setup] ******************************************************************
ok: [172.16.1.192]
ok: [172.16.1.217]

TASK [ensure apache is at the latest version] *********************************
ok: [172.16.1.192]
ok: [172.16.1.217]

TASK [ensure httpd is started] ************************************************
changed: [172.16.1.192]
changed: [172.16.1.217]

TASK [ensure mysql is at the latest version] **********************************
changed: [172.16.1.192]
changed: [172.16.1.217]

TASK [ensure mysqld is started] ***********************************************
changed: [172.16.1.192]
changed: [172.16.1.217]

PLAY RECAP ********************************************************************
172.16.1.192             : ok=5    changed=3    unreachable=0    failed=0
172.16.1.217             : ok=5    changed=3    unreachable=0    failed=0
```

4.2.2. Exemple de préparation d'un noeud mysql

Dans ce playbook, deux serveurs Cloud vont être préparés pour devenir des serveurs Multi-Maître MySQL (voir cours MySQL Multi-Maître).

Le playbook utilise :

- Des variables ;
- Ajoute des lignes dans le fichier /etc/hosts ;
- Installe et démarre MariaDB ;
- Créer une base de données, un utilisateur et lui donne tous les droits sur les bases de données.

```
---

- hosts: aws
  remote_user: ec2-user
  become: yes
  vars:
    mysqlpackage: "mariadb-server,MySQL-python"
    mysqlservice: "mariadb"
    mysql_port: "3306"
    dbuser: "synchro"
    dbname: "mabase"
    upassword: "M!rro!r"

  tasks:
    - name: configurer le noeud 1 dans /etc/hosts
      lineinfile:
              dest: /etc/hosts
              line: "13.59.197.48 mirroir1.local.lan mirroir1"
              state: present
    - name: configurer le noeud 2 dans /etc/hosts
      lineinfile:
              dest: /etc/hosts
              line: "52.14.125.109 mirroir2.local.lan mirroir2"
              state: present
    - name: mariadb installe et a jour
      yum: name="{{ mysqlpackage }}" state=latest
    - name: mariadb est demarre
      service: name="{{ mysqlservice }}" state=started
    - name: creer la base de donnee
      mysql_db: name="{{ dbname }}" state=present
    - name: creer un utilisateur
      mysql_user: name="{{ dbuser }}" password="{{ upassword }}" priv=*.*:ALL host='%'
state=present
    - name: restart mariadb
      service: name="{{ mysqlservice }}" state=restarted
...
```