

Projet Ansible v1

2017/2018

1. Installation et configuration sous CentOS 7

a. Configuration réseau liée au réseau de l'école

Modifier « */etc/profile* » et ajouter :

```
MY_PROXY="http://129.194.185.57:3128/"
HTTP_PROXY=$MY_PROXY
HTTPS_PROXY=$MY_PROXY
FTP_PROXY=$MY_PROXY
http_proxy=$MY_PROXY
https_proxy=$MY_PROXY
ftp_proxy=$MY_PROXY
export HTTP_PROXY HTTPS_PROXY FTP_PROXY http_proxy https_proxy ftp_proxy
```

Exécuter la commande **source /etc/profile**

Modifier « */etc/yum.conf* » et ajouter :

```
proxy = http://129.194.185.57:3128/
```

b. Update et installation

Update :

```
Sudo yum update
```

Installation :

```
Sudo yum install ansible
```

Si problème de dépôt :

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

c. Configuration

Ouvrir le fichier hosts dans « *vi /etc/ansible/hosts* » et ajouter :

```
[group_name]
alias ansible_ssh_host=your_server_ip
```

```
[servers]
host1 ansible_ssh_host= ip address
host2 ansible_ssh_host= ip address
host3 ansible_ssh_host= ip address
```

Exemple:

Server_web ansible_ssh_host = 10.0.0.2

Les hôtes peuvent être dans des groupes multiples et les groupes peuvent configurer des paramètres pour tous leurs membres.

Ensuite faut créer un dossier groupe qui va contenir nos configurations des groupes créés dans Hosts :

```
sudo mkdir /etc/ansible/group_vars
```

Puis créer un fichier au nom du groupe (ici ces « servers ») et l'ouvrir :

```
sudo vi /etc/ansible/group_vars/servers
```

Et intégrer les informations pour la connexion :

```
---
ansible_ssh_user: « nom d'utilisateur pour le SSH »
```

Ensuite il faut générer des clés pour le SSH entre le(s) client(s) et le serveur :

```
ssh-add //add the ssh key to the agent
ssh-keygen -t rsa -C "user@ip_ansible"
ssh-copy id user@ipducient
```

Dans notre cas :

```
ssh-keygen -t rsa -C "ansible@ 10.194.184.190"
ssh-copy id root@ 10.194.184.191
```

Puis lancer le test :

```
ansible -m ping « nom du groupe à ping »
```

Si ça marche :

```
host1 | SUCCESS => {
  "changed": false,
  "ping": "pong"
}
```

Ou si sa marche pas :

```
host1 | UNREACHABLE! => {  
  "changed": false,  
  "msg": "Failed to connect to the host via ssh.",  
  "unreachable": true  
}
```

2. Création et exécution de playbooks

a. Commande pour lancer un playbooks :

```
ansible-playbook chemindufichier/name.yml
```

Avec variable en paramètre:

```
ansible-playbook chemindufichier/name.yml -e "nomvariable=valueofvariable"
```

-e = --extra-vars

Exemple :

```
ansible-playbook /playbooks/ping.yml
```

b. Création et configuration d'un fichier YML

Création et ouverture (exemple avec lancement d'un ping) :

```
vi /playbooks/ping.yml
```

Et le contenu :

```
---
- hosts: Clients
  remote_user: root

  tasks:
    - ping:
```

hosts : on doit indiquer le groupe ou la machine à qui on veut faire exécuter notre futur taches/services

remote_user : indiqué sous qu'elle utilisateur nous voulons lancer la prochaine taches/services

tasks : pour indiquer que la prochaine ligne sera une taches a faire

-ping : indique le nom de la commande à lancer

On peut aussi démarrer, arrêter, relancer un service (par exemple apache)

```
- service:
  name: httpd
  state: started
```

service : pour indiquer qu'on veut travailler sur un service

name : le nom du service qu'on veut toucher

state : indique l'état futur (started, stopped, restarted, reloaded)

Exemple d'utilisation de variable avec récupération de la valeur en paramètre de la commande :

```
---
vars:
  Choice: "{{ choice }}"

tasks:
- include: Web.yml
  when: 1
- include: update_web.yml
  when: 2
```

3. Roles

a) A quoi ça sert (extrait de buzut.fr)

Les rôles représentent une manière d'abstraire les directives includes. C'est en quelque sorte une couche d'abstraction. Grâce aux rôles, il n'est plus utile de préciser les divers includes dans le playbook, ni les paths des fichiers de variables etc. Le playbook n'a qu'à lister les différents rôles à appliquer.

En outre, depuis les tasks du rôle, l'ensemble des chemins sont relatifs. Inutile donc de préciser l'intégralité du path lors d'un copy, template ou d'une tâche. Le nom du fichier suffit, Ansible s'occupe du reste.

ansible-galaxy prépare l'arborescence d'un rôle vide. Nos playbooks seront à la racine de notre dossier tandis que les rôles seront dans roles. Ainsi, lorsque nous appellerons un rôle depuis un playbook, sans avoir besoin de préciser autre chose que son nom, Ansible saura où chercher.

Extrait de buzut.fr :

Defaults : les variables par défaut qui seront à disposition du rôle.

Vars : Variables à disposition du rôle cependant elles ont vocation à être modifiées par l'utilisateur et elles prennent le dessus sur celle du dossier « defaults » si elles sont renseignées.

Tasks : ici on renseigne nos tâches (comme dans un playbooks normal)

Meta : Sert à renseigner les dépendances liées à nos rôles (ssl, et etc).

README : renseigne sur comment utilisé les rôles, variables à définir et etc.

b) Création et configuration pour httpd (CentOS 7)

Il y a un dossier *roles* dans */etc/ansible/* donc aller dans le dossier */etc/ansible/roles/*.

Puis crée notre rôle qui sera pour httpd :

```
ansible-galaxy init httpd
```

Cela créé plusieurs dossiers dans httpd: defaults, handlers, meta, README.md, tasks, test, vars.

Pour configuré des taches il faut éditer le main.yml dans *httpd/tasks/* :

```
sudo nano ./httpd/tasks/main.yml
```

```
---
# tasks file for httpd
- name: add repository epel release
  yum:
    name: epel-release
    state: latest

- name: install httpd
  yum: name={{ item }} state=latest
  with_items:
    - '*'
    - nano
    - httpd
    - php
    - php-mysql
  notify:
    - Start httpd
```

notify est pour démarrer un service qui se situe dans le handlers (*httpd/handlers/*) :

```
---
# handlers file for httpd
- name: Start httpd
  service:
    name: httpd
    state: started

- name: Stop httpd
  service:
    name: httpd
    state: stopped

- name: reload httpd
  service:
    name: httpd
    state: reloaded
```

Le nom donner a "name" est le nom qu'on va utiliser pour appeler le démarrage, stop,... du/des service(s).

Ici on utilisera pas les autres fichier qui sont dans « meta , vars , ... ».

Puis il faut éditer le fichier `/etc/ansible/ansible.cfg` et ajouter cette ligne à la fin :

```
roles_path = /etc/Ansible/roles/
```

En dehors du dossier rôles, il faut créer le main qui va lancer le rôle :

```
sudo nano server.yml
```

Puis le remplir :

```
---
- hosts: localhost
  remote_user: root
  roles:
    - httpd
```

Hosts et remote_user (cf Création et configuration d'un fichier YML)

roles : et là où on appelle le rôle qu'on veut utiliser

Et pour le lancer :

```
ansible-playbook -s server.yml
```

c) Choix de service et système d'exploitation différent

On peut indiquer en paramètres le service qu'on veut installer et détecter le système d'exploitation qui est utilisé pour appeler le bon rôle et installer avec les bonnes commandes.

```
vars:
  service: "service"

tasks:

- include_role:
  name: Web
  when:
    - ansible_os_family == "RedHat"
    - service == "Web"

- include_role:
  name: apache
  when:
    - ansible_os_family == "Debian"
    - service == "Web"
```

Include_role : permet d'appeler un rôle

when : une condition «if» qui exécutera celui qui y a juste au-dessus (ici exécution du rôle) si le(s) conditions sont respectées.

ansible_os_family : vas vérifier si le serveur a bien fait partie de la famille indiqué

On peut aussi faire pour une version précise de l'OS :

```
when: ansible_distribution == 'CentOS' and ansible_distribution_version == '7.0'
```

Ou par release :

```
when: ansible_distribution == 'Ubuntu' and ansible_distribution_release == 'precise'
```

Pour Apache (Debian), seul les commandes vont changer pour installer des paquets :

```
#Configuration required for based OS on Debian
- name: update all package to the latest version (ubuntu)
  apt:
    upgrade: dist

- name: build-essential
  apt:
    name: build-essential
    state: latest

- name: Apache 2 installation
  apt: name={{ item }} state=latest
  with_items:
    - "apache2"
    - php5-common
    - libapache2-mod-php5
    - php5-cli

- service: name=apache2 state=started enabled=yes
```


Bibliographie :

<https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-ansible-on-centos-7>

<https://github.com/ansible/ansible/issues/19584>

https://docs.ansible.com/ansible/latest/yum_module.html

https://docs.ansible.com/ansible/latest/service_module.html

https://docs.ansible.com/ansible/latest/apt_module.html

https://docs.ansible.com/ansible/latest/apt_repository_module.html

<https://serversforhackers.com/c/an-ansible-tutorial>

<https://buzut.fr/tirer-toute-puissance-dans-ansible-roles/>