Projet Ansible v1

2017/2018

1. **Installation et configuration sous centOS 7**
   1. **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/ |

* 1. **Update et installation**

Update :

|  |
| --- |
| Sudo yum update |

Installation :

|  |
| --- |
| Sudo yum install ansible |

Si problème de dépôt :

|  |
| --- |
| sudo yum install epel-release |

* 1. **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 crée un dossier groupe qui vas contenir nos configuration des groupes crée dans Hosts :

|  |
| --- |
| sudo mkdir /etc/ansible/group\_vars |

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

|  |
| --- |
| sudo vi /etc/ansible/group\_vars/servers |

Et intégré les informations pour la connexion :

|  |
| --- |
| ---  ansible\_ssh\_user: « nom d’user pour le SSH » |

Ensuite il faut générer des clefs 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@ipduclient |

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 a ping » |

Si sa a 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  } |

1. **Création et exécution de playbooks** 
   1. **Commande pour lancer un playbooks :**

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

Avec variable en paramètre:

|  |
| --- |
| ansible-playbool chemindufichier/name.yml –e “nomvariable=valueofvariable” |

-e = --extra-vars

Exemple :

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

* 1. **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 |

1. **Roles**
2. **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 taches (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.

1. **Création et configuration pour httpd (CentOS 7)**

Il y a un dossier *roles* dans */etc/Ansibe/* 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 celle ligne à la fin :

|  |
| --- |
| roles\_path = /etc/Ansible/roles/ |

En dehors du dossier rôles, il faut crée le main qui vas 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 ou on appelle le rôle qu’on veut utiliser

Et pour le lancer :

|  |
| --- |
| ansible-playbook –s server.yml |

1. **Choix de service et système d’exploitation diffèrent**

On peut indiquer en paramètres le service qu’on veut installer et détecter 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 respectés.

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-dansible-roles/>