

ANSIBLE ADVANCED



SOMMAIRE

Ansible

- 1.Les facts
- 2.Les variables magiques
- 3.Un peu plus loin dans les variables
- 4.Les rôles
- 5.Sécurité: Ansible vault
- 6.Les collections



Ansible - Les Facts

1.1 Définitions

Les facts sont des variables d'un type particulier

Ce sont des variables contenant des informations qui proviennent directement des serveurs distants

On peut utiliser ce type de variable pour par exemple connaître le système d'exploitation de chacun de vos serveurs et faire lancer certaines tâches de manière conditionnelle.



Ansible - Les Facts

1.2 Utilisation

Il a plusieurs façons d'obtenir les facts
via un playbook

```
- name: Print all available facts
```

```
  ansible.builtin.debug:
```

```
    var: ansible_facts
```



Ansible - Les Facts

1.2 Utilisation

via la ligne de commande

```
ansible <hostname> -m ansible.builtin.setup
```



Ansible - Les Facts

1.2 Utilisation

Le résultat de ces commandes ressemble à ça:

```
{  
  "ansible_all_ipv4_addresses": [  
    "REDACTED IP ADDRESS"  
  ],  
  "ansible_all_ipv6_addresses": [  
    "REDACTED IPV6 ADDRESS"  
  ],  
}
```



Ansible - Les Facts

1.2 Utilisation

Un exemple d'utilisation

```
- name: install apache and php last version for (Debian os family)
  apt:
    name: ['apache2', 'php', 'php-mysql']
    state: present
    update_cache: yes
  when: ansible_facts['os_family'] == "Debian"
```



Ansible - Les Facts

1.2 Utilisation

Un exemple d'utilisation

```
- name: install apache and php last version for (RedHat os family)
yum:
  name: ['httpd', 'php', 'php-mysqlnd']
  state: present
  update_cache: yes
when: ansible_facts['os_family'] == "RedHat"
```



Ansible - Les Facts

1.2 Utilisation

Par défaut les facts sont gardés en mémoire, cependant il est possible de les rendre persistants afin d'y accéder de manière répétée , ça veut dire quoi ?
Depuis un playbook vous pouvez accéder aux facts de tous vos hosts

```
{{ hostvars['asdf.example.com']['ansible_facts']['os_family'] }}
```



Ansible - Les Facts

1.2 Utilisation

Il est également possible de désactiver les facts principalement pour des raisons de performances

```
- hosts: whatever  
  gather_facts: no
```



Ansible - Les Facts

1.2 Utilisation

Bon à savoir par défaut ansible rassemble les facts en début de lancement de playbook



Ansible - Les variables magiques

2.1 Définition

Les variables magiques sont l'ensemble des variables liés aux opérations d'ansible

Les noms de ces variables sont réservés et ne peuvent donc pas être utilisée par d'autres variables

```
{{ hostvars['test.example.com']['ansible_facts']['distribution'] }}
```



Ansible - Les variables magiques

2.2 Utilisation

Comme pour tout le reste

```
{% for host in groups['app_servers'] %}  
    {{ hostvars[host]['ansible_facts']['eth0']['ipv4']['address'] }}  
{% endfor %}
```

ansible_play_hosts, role_path, ansible_check_mode



Ansible - Les variables advanced

3.1 Utilisation

On a déjà vu comment fonctionne les variables et comment les utiliser , on va maintenant voir que les variables peuvent s'utiliser à plusieurs endroits différents

on imagine que l'on a la variable `ntp_server`

- si l'on veut que cette variable soit disponible pour tous nos hosts

```
# file: /etc/ansible/group_vars/all
```

```
# this is the site wide default
```

```
ntp_server: default-time.example.com
```



Ansible - Les variables advanced

3.1 Utilisation

- si l'on veut que cette variable soit disponible pour tous les hosts de type webservers

```
# file: /etc/ansible/group_vars/webrowsers
```

```
ntp_server: boston-time.example.com
```



Ansible - Les variables advanced

3.1 Utilisation

- si l'on veut que cette variable soit disponible pour un seul host

```
# file: /etc/ansible/host_vars/xyz.boston.example.com
```

```
ntp_server: override.example.com
```



Ansible - Les variables advanced

3.2 En pratique

Choisissez un package à installer (wget , telnet, tcpdump, ldapsearch etc ...) et créer un playbook qui installe ce package sur les OS de type “Debian” et “RedHat”

Tester votre playbook sur les 2 types d’OS si c’est possible



Ansible - Les variables advanced

3.2 En pratique / Bonus

Créer votre propre fact(dynamique) et remonté le via un playbook

```
ansible.builtin.set_fact
```



Ansible - Les rôles

4.1 Définition

Les rôles vous permettent de charger automatiquement des variables, des fichiers, des tâches, des gestionnaires et d'autres artefacts Ansible associés en fonction d'une structure de fichiers connue. Après avoir regroupé votre contenu dans des rôles, vous pouvez facilement les réutiliser et les partager avec d'autres utilisateurs.

En résumé les rôles vous permettent de réutiliser votre code dans des playbooks différents



Ansible - Les rôles

4.2 Utilisation

Un rôle ansible à une structure bien définie, pour que le rôle existe il faut au moins qu'il possède un sous-répertoire



Ansible - Les rôles

4.2 Utilisation

```
# playbooks  
  
site.yml  
  
webservers.yml  
  
fooservers.yml  
  
roles/  
  
    common/  
  
        tasks/  
  
        handlers/  
  
        library/  
  
        files/  
  
        templates/  
  
        vars/  
  
        defaults/  
  
        meta/  
  
webservers/  
  
    tasks/
```



Ansible - Les rôles

4.2 Utilisation

Les répertoires

- tasks: l'ensemble des tâches disponible pour un host
- defaults: les variables par défaut pour le rôle
- vars: les autres variables du rôle (celle que l'on ne pourra pas surcharger facilement)
- templates: les templates à déployer pour le rôle



Ansible - Les rôles

4.2 Utilisation

Les tâches

- par défaut ansible prend le fichier main.yml se trouvant dans ce répertoire mais il est possible de le splitter en plusieurs fichiers



Ansible - Les rôles

4.2 Utilisation

```
# roles/example/tasks/main.yml
```

```
- name: Install the correct web server for RHEL
  import_tasks: redhat.yml
  when: ansible_facts['os_family']|lower == 'redhat'
```

```
- name: Install the correct web server for Debian
  import_tasks: debian.yml
  when: ansible_facts['os_family']|lower == 'debian'
```

```
# roles/example/tasks/redhat.yml
```

```
- name: Install web server
  ansible.builtin.yum:
    name: "httpd"
    state: present
```

```
# roles/example/tasks/debian.yml
```

```
- name: Install web server
  ansible.builtin.apt:
    name: "apache2"
    state: present
```



Ansible - Les rôles

4.2 Utilisation

Comment on utilise un rôle dans un playbook

```
---  
- hosts: webservers  
  roles:  
    - common  
    - webservers
```



Ansible - Les rôles

4.2 Comment qu'on fait vraiment

on crée un répertoire qui porte le nom du rôle

```
mkdir assruser
```

on crée les sous-répertoires nécessaires à la création du rôle

```
mkdir assruser/tasks assruser/vars assruser/templates assruser/files
```



Ansible - Les rôles

4.2 Comment qu'on fait vraiment

on créé dans le repertoire tasks un fichier main.yml

```
---
```

```
- name: Include install tasks
```

```
  include_tasks:
```

```
    file: install.yml
```

```
  when: install is defined
```



Ansible - Les rôles

4.2 Comment qu'on fait vraiment

on crée dans le repertoire tasks un fichier install.yml qui contient notre code

```
---  
- name: Install package  
  apt:  
    dpkg:  
      - curl
```



Ansible - Les rôles

4.2 Comment qu'on fait vraiment

- Maintenant que le rôle est créé on va créer un playbook qui appelle ce rôle

```
ole > playbooks > dagger > ! install.yml
```

```
---
- hosts: dagger
  vars:
    install: true
  tasks:
    - name: Info debug
      ansible.builtin.debug:
        msg: "Install dagger"
    - name: Include dagger role
      include_role:
        name: dagger
```



Ansible - Les rôles

4.2 Pratique

Créer un rôle qui va gérer un utilisateur

- ajout d'un user
- suppression d'un user
- changement de son mot passe



Ansible - Les rôles

4.2 Pratique/Bonus

Créer une tâche supplémentaire dans ce rôle qui supprime l'utilisateur si il existe si et seulement si le hostname du serveur match avec “front”



Ansible - Ansible vault

5.1 Définition

Un vault est un coffre-fort, c'est un espace de stockage chiffré où l'on conserve les données sensibles comme les mots de passes, les certificats ssl etc

En règle générale, on aime bien versionner nos fichiers (donc nos playbooks) et comme les playbooks contiennent des identifiants il est impératif de les chiffrer, ansible vault est fait pour ça



Ansible - Ansible vault

5.2 Utilisation

via une variable chiffrée dans un playbook

```
ansible-vault encrypt string 'secret-text' --name 'secret'
```



Ansible - Ansible vault

5.2 Utilisation

Résultat :

```
secret: !vault |
  $ANSIBLE_VAULT;1.1;AES256
  33616163316438306663303330376334363862343430363432343536313835323132353939376438
  34393333730353532346630626133666434363932303133650a343137353735363138646536633432
  39303765306633643830353238646433356230623537363466373438323931353763666537386163
  3062303334343830370a3339393466323461343130633631363630383562663038353633331376631
  3664
```



Ansible - Ansible vault

5.2 Utilisation

Ansible demandera ensuite le mdp pour chiffrer , vous pouvez le mettre directement dans votre var

```
---
- hosts: localhost
  vars:
    - secret: !vault |
      $ANSIBLE_VAULT;1.1;AES256
      33616163316438306663303330376334363862343430363432343536313835323132353939376438
      3439333730353532346630626133666434363932303133650a343137353735363138646536633432
      39303765306633643830353238646433356230623537363466373438323931353763666537386163
      3062303334343830370a333939346632346134313063363136363038356266303835363331376631
      3664

  tasks:
    - debug: var=secret
    - fail:
      when: 1 == 1
```

Copier



Ansible - Ansible vault

5.2 Utilisation

Ensuite vous devez exécuter votre playbook avec l'option `--ask-vault-pass`

```
ansible-playbook playbook.yml --ask-vault-pass
```



Ansible - Ansible vault

5.3 Pratique

utilisez ansible vault pour chiffrer une de vos variable et lancer votre playbook



Ansible - Ansible vault

5.3 Pratique/Bonus

utilisez ansible vault pour chiffrer plusieurs de vos variable et trouver un moyen de ne pas demander le mot de passe a chaque lancement du playbook



Ansible - Les collections

6.1 Définitions

Les collections sont un simple format de distribution de contenu pour Ansible Galaxy.

Dans les faits, une collection est une grosse archive compressée qui contient votre contenu dans une arborescence stricte :

```
collection/  
├── docs/  
├── galaxy.yml  
├── plugins/  
│   ├── modules/  
│   │   └── module1.py  
│   ├── inventory/  
│   └── .../  
├── README.md  
├── roles/  
│   ├── role1/  
│   ├── role2/  
│   └── .../  
└── playbooks/
```



Ansible - Les collections

6.1 Définitions

Comme on peut le voir on retrouve des notions que l'on connaît

- les roles
- les playbooks

et une nouvelle notion de plugins

- . plugins/modules : pour vos modules.
- . plugins/inventory : pour vos plugins d'inventaire.
- . plugins/filter : pour vos filtres.
- . plugins/module_utils : pour le code commun à vos plugins !



Ansible - Les collections

6.1 Définitions

- Une collection vous permettra donc de créer une suite de composants utilisables avec Ansible.
- Les filtres et modules ne seront plus liés à un rôle mais utilisables n'importe où dès le moment où la collection sera installée.
- La présence du répertoire plugins/module_utils permettra également de factoriser le code (Python vraisemblablement) entre les différents composants de la collection ce qui facilitera la maintenance pour l'auteur de la collection.



Ansible - Les collections

6.1 Pratiques

ansible-galaxy est l'exécutable qui permet d'installer des collections qui sont publiés sur le site
ansible-galaxy

```
ansible-galaxy collection install my_namespace.my_collection
```



Ansible - Les collections

6.1 Pratiques

on peut également installer les collections de manière plus sécurisée en vérifiant la signature

```
gpg --import --no-default-keyring --keyring ~/.ansible/pubring.kbx my-public-key.asc
```

```
ansible-galaxy collection install my_namespace.my_collection --keyring  
~/.ansible/pubring.kbx
```



Ansible - Les collections

6.2 Création

On va créer une collection pour cela on va créer un répertoire sur notre server ansible-server
les scripts sont présents sur le repo github learning

```
mkdir ansible-collections
```

On se placera désormais dans ce répertoire pour lancer toutes les commandes



Ansible - Les collections

6.2 Création

On va initialiser notre collection

```
ansible-galaxy collection init asr.my collection
```

ça va nous créer tous les répertoires et fichiers nécessaires à notre collection

Le formalisme est important il permettra de pouvoir les publier sur ansible



Ansible - Les collections

6.2 Création

On va créer un rôle myip qui affichera l'ip de notre vm

```
ansible-galaxy role init myip
```

ça va nous créer tous les répertoires et fichiers nécessaires à notre collection

Le formalisme est important il permettra de pouvoir les publier sur ansible



Ansible - Les collections

6.2 Création

On va ensuite modifier la tâche du fichier main.yml pour qu'elle affiche l'IP

```
# /etc/ansible/roles/ansible_collections/ansible/ansible/tasks/main.yml
---

- name: mon debug
  debug:
    msg: "{{ ansible_default_ipv4.address }}"

...
```



Ansible - Les collections

6.2 Création

Ajouter un filtre

```
from __future__ import (absolute_import, division, print_function)
__metaclass__ = type

def split_ip(value):
    return value.split(".")

class FilterModule(object):

    def filters(self):
        return {
            'split_ip': split_ip
        }
```



Ansible - Les collections

6.2 Création

Ce fichier est un exemple de filtre, un filtre permet d'étendre les fonctionnalités d'ansible
La fonction créée dans le filtre (en python donc) permet d'être utilisée dans un playbook

```
-- --  
- name: mon debug  
  debug:  
    msg: "{{ ansible_default_ipv4.address | rrey.my_collection.split_ip }}"  
...  

```



Ansible - Les collections

6.2 Création

Maintenant qu'on a créé une collection on va configurer ansible pour que l'on puisse l'utiliser
On pourrait la publier sur le site ansible-galaxy mais ça ne servirait à rien et ce n'est pas le but ici
On va donc aller dans notre fichier ansible.cfg et changer la conf pour la key collections_path

```
# ansible.cfg  
[defaults]  
collections_paths = ~/.ansible/collections:/usr/share/ansible/collections:/Users/remi.rey
```



Ansible - Les collections

6.3

A vous de jouer

Maintenant que l'on a créé une collection et configuré ansible pour pouvoir l'utiliser on va créer un playbook qui utilise et appelle cette collection

