Support pédagogique Ansible - 2

Sommaire

- La documentation avec ansible-doc
- Cas pratique avec l'installation de Apache
- YAML
- Les Playbooks
- Vérification des playbooks avec ansible-lint
- Les Variables

La documentation avec ansible-doc

> La documentation de Ansible peut être consultée avec l'outil `ansible-doc`

- Pour connaître les options à utiliser avec le module `yum`

```
$ ansible-doc yum
```

- En plus des options, des exemples sont donnés :

```
[...]
EXAMPLES:
- name: install the latest version of Apache
  yum:
    name: httpd
    state: latest

- name: remove the Apache package
  yum:
    name: httpd
    state: absent
```

Cas pratique avec l'installation de Apache

- Installation du package `Apache` (`-m yum` et l'option `name=httpd`)

```
$ ansible -i inventaire -b -m yum -a name=httpd all --ask-become-pass
```

```
SUDO password:
centos | SUCCESS => {
    "changed": true,
    "msg": "",
    "rc": 0,
    "results":
```

- Contrôle si `Apache` est installé

```
$ systemctl status httpd
```

```
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service;
   disabled; vendor preset: disabled)
   Active: inactive (dead)
```

- Activation et démarrage du service `Apache` (`state=started` et `enabled=yes`) ;

```
$ ansible -i inventaire -b -m service \
                        -a "name=httpd state=started enabled=yes" all -K
```

```
SUDO password:
centos | SUCCESS => {
    "changed": true,
    "enabled": true,
    "name": "httpd",
    "state": "started",
    "status":
```

- Controle du résultat sur le serveur

```
$ systemctl status httpd
```

```
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service;
   enabled; vendor preset: disabled)
   Active: active (running) since jeu. 2018-06-21 22:52:15 CEST; 2min 29s ago
```

> Une fois le serveur Apache installé, nous allons copier un fichier à la racine du site en utilisant cette fois le module `copy`

- Copie d'un fichier (`www.html`) à la racine du serveur, avec les options du module `copy`
  - l'emplacement du fichier source : `src=www.html`
  - le répertoire destination : `dest=/var/www/html`
  - le propriétaire du fichier : `owner=apache`
  - le groupe du fichier : `group=apache`
  - le mode de protection du fichier : `mode=644`
- Lancement des instructions ad hoc

```
$ ansible -i inventaire -b -m copy \
        -a "src=www.html owner=apache group=apache dest=/var/www/html" all -K
```

```
centos | SUCCESS => {
    "changed": true,
    "checksum": "8f3f87ae883b60c9bf50c900bb414c039e900827",
    "dest": "/var/www/html/www.html",
    "gid": 48,
    "group": "apache",
    "md5sum": "5e6f102c864ed5bd9eacdad977d1e5b7",
    "mode": "0644",
    "owner": "apache",
    "secontext": "system_u:object_r:httpd_sys_content_t:s0",
    "size": 21,
    "src": "/home/alex/.ansible/tmp/ansible-tmp-1529614972.89-
    143711028043810/source",
    "state": "file",
    "uid": 48
}
```

- Le résultat visible sur le serveur Apache

```
$ curl centos/www.html
```

```
Ansible <> Bonjour !
```

YAML - Yet Another Markup Language

> YAML permet d'écrire des **structures de données** qui peuvent être spécifiées sous la forme de **listes** ou sous la forme de **dictionnaires**.

Un nom de variable valide devra impérativement être composé d'une suite de caractères **alphanumériques** et/ou de **tirets bas** et le premier caractère devra toujours être un caractère **alpha** ou un **tiret bas** :

- Bien : `_42, var1, a_b`
- Pas bien : `42a, a-b, a%b`

> Un fichier YAML type

```yaml
---
# Un fichier yaml démarre par les 3 tirets ci-dessus
# Déclaration simple
chaîne_simple: "Une chaîne simple"
# Ici _42 va contenir un entier :
_42: 42
# _33 va contenir un chiffre à virgule :
_33: 33.333
```

Les tableaux

```yaml
a:
  - 1
  - 2
  - "trois"
```

> ou alors, façon `JSON` avec les `[]`

```
# Version compacte
a: [ 1, 2, "trois" ]

# Version très compacte
a: [1,2,"trois"]

# Version invalide (manque l'espace après ':')
a:[1,2,"trois"]
```

> Contenu de la variable `a` est de `a[0]` à `a[2]`

Les structures clé/valeur

```
utilisateur1:
  nom: martin
  prenom: pierre
  date_de_naissance:
    jour: 6
    mois: 10
    annee: 1977
```

> ou

```
utilisateur2: { nom: martin, prenom: pierre }
```

> Contenu de la variable `nom` de `utilisateur2` est `utilisateur2.nom`

Tableau de tables de hachage

> Un mixte de variables, tableaux, structures clé/valeur

```
liste_utilisateurs:
 - nom: martin
   prenom: jean
 - nom: martin
   prenom: anne
```

> Contenu de la variable `prenom` est `liste_utilisateurs[nom.prenom]`

Les Playbooks

> Un playbook est un fichier au format `YAML`. Ce dernier va donner une liste d'instructions. Ces instructions sont passées à Ansible dans l'ordre de leur déclaration. L'avantage par rapport au mode ad hoc est que tout est écrit dans un fichier, y compris l'enchaînement des opérations.

Installation et activation du service Apache avec un Playbook

- Le fichier de configuration, `install-apache.yml`

```yaml
---
# name : le nom du playbook (apporte de la clarté au code)
- name: "Apache Installation"
# hosts : la liste des machines sur lesquels nous allons travailler
  hosts: all
# tasks : une liste d'instructions à dérouler
  tasks:
    - name: "Install apache package"
# yum : module pour l'installation du package
      yum:
# Les differentes options
        name: "httpd"
        state: "present"
    - name: "Start apache service"
      service:
        name: "httpd"
        state: "started"
        enabled: yes
    - name: "Copy www.html"
      copy:
        src: "www.html"
        dest: "/var/www/html"
        owner: "apache"
        group: "apache"
```

Lancement d’un playbook avec la commande `ansible-playbook`

```
$ ansible-playbook -b -K -i inventaire install-apache.yml
```

- l’inventaire `-i inventaire`
- le playbook à lancer `install-apache.yml`
- l’option `--become` ou `-b`
- l'option `--ask-become-pass` ou `-K`

- Résultat de la sortie (callback)

```
PLAY [Apache Installation] ***************************************************

TASK [Gathering Facts]  ******************************************************
ok: [centos]

TASK [Install apache package] ************************************************
ok: [centos]

TASK [Start apache service]  *************************************************
ok: [centos]

TASK [Copy www.html]  ********************************************************
ok: [centos]

PLAY RECAP  ******************************************************************
centos                     : ok=4    changed=0    unreachable=0    failed=0
```

> Tous les actions sont validées (`ok=4`) puisque elles ont deja été réalisées précédement

Installer Python sur les serveurs distants

> Dans certain cas, Python n'est pas installé sur le serveur distant ou le chemin de l’interpréteur n'est pas habituel.

- Depuis sa version **2.2**, Ansible est compatible avec Python **3** (l'adaptation des modules sont en cours)
- Au niveau des versions minimums, vous devez disposer d’une version **3.5** pour Python **3** et **2.6** pour Python **2**

- Le `playbook` pour installer `Python` au minimum dans le fichier `install-python.yml`

```
---
# name : le nom du playbook (apporte de la clarté au code)
- name: "install python2"
# hosts : la liste des machines sur lesquels nous allons travailler
  hosts: all
# Les caractéristiques des serveurs ne sont pas demandées
  gather_facts: no
# tasks : une liste d'instructions à dérouler
  tasks:
    - name: "Python2 installation using apt"
# raw : module pour faire une action sans faire appel au module (python)
      raw: apt install -y python-minimal
      args:
        creates: "/usr/bin/python"
```

> Attention, cet exemple est valable uniquement pour les environnements Debian. Il faut adapter pour Centos

- Exécution du `playbook`

```
$ ansible-playbook -i inventaire install-python.yml
```

- Dans certains cas, le chemin de l'interpréteur Python doit être reprécisé grâce à la variable `ansible_python_interpreter` qui est ici placée dans le fichier `inventaire`.

```
[srvp3:vars]
ansible_python_interpreter=/usr/bin/python3
```

Vérification des playbooks avec ansible-lint

- Installer le paquet

```
pip install ansible-lint
```

- Lance la verification du fichier `install-apache.yml`

```
$ ansible-lint install-apache.yml
```

```
[ANSIBLE0002] Trailing whitespace
install-apache.yml:11
# Les differentes options
```

- Ouvre le fichier à modifier avec l'option `list` pour faire apparaître les espaces

```
$ vi +11 "+set list nu" install-apache.yml
```

```
1 ---$
2 # name : le nom du playbook (apporte de la clarté au code)$
3 - name: "Apache Installation"$
4 # hosts : la liste des machines sur lesquels nous allons travailler$
5   hosts: all$
6 # tasks : une liste d'instructions à dérouler$
7   tasks:$
8     - name: "Install apache package"$
9 # yum : module pour l'installation du package$
10      yum:$
11 # Les différentes options $
12        name: "httpd"$
13        state: "present"$
14    - name: "Start apache service"$
15      service:$
16        name: "httpd"$
17        state: "started"$
18        enabled: yes$
19    - name: "Copy www.html"$
20      copy:$
21        src: "www.html"$
22        dest: "/var/www/html"$
23        owner: "apache"$
24        group: "apache"$
```

Les Variables

> Il est parfois utile de définir des variables en fonction du contexte d'exécution

- Soit directement dans le fichier inventaire

```
[lamp]
web apache_url=rec.wiki.localdomain mysql_user_password=MyPassWord!
```

- Soit dans des répertoires `group_vars` et `host_vars`, avec un fichier par serveur ou groupe de serveurs au format `YAML` ou `JSON`
- Soit en mode ad hoc avec l'option `-e`

```
$ ansible -e variable=valeur -e @fichier-variables.yml \
          -m debug -a var=variable,varfile localhost
```

```
localhost | SUCCESS => {
    "variable,varfile": "(u'valeur', u'une variable')"
}
```

Hiérarchie et priorité des variables

> Par ordre de priorité

1. variables se trouvant dans un fichier YAML (`-e @fichier.yml`)
2. variables directement passées à Ansible (`-e variable=valeur`)
3. variables de la machine au niveau du fichier `host_vars`
4. variables de la machine au niveau du fichier `host`
5. variables du groupe dans les fichiers `group_vars`
6. variables du groupe dans le fichier `host`

> Il y encore d'autres variables qui peuvent être trouvés dans les `Roles` ou dans les variable d'environnement `facts` ....

- Un exemple avec des variables dans le `playbook`

```
- hosts: all
  remote_user: root
  vars:
    favcolor: blue
  vars_files:
    - /vars/external_vars.yml
```

> Théoriquement, il faudrait utiliser les variables au niveau des groupes dans le répertoire `group_vars`. Le fichier `host`, ne doit contenir que des déclarations de serveurs.

> playbooks/host_vars/quebec.example.com

> playbooks/group_vars/production

```
db_primary_host: rhodeisland.example.com

rabbitmq_host:pennsylvania.example.com
```

ou

```
db:
  user: widgetuser
  password: pFmMxcyD;Fc6)6
rabbitmq:
  host: pennsylvania.example.com
  port: 5672
```

> playbooks/group_vars/production/db

```
db_primary_host: rhodeisland.example.com
db_primary_port=5432
db_replica_host: virginia.example.com
```

Test de la hiérachie des variables

1. Créer un playbook qui affiche une variable **prio** définie dans la section `vars:` du Playbook (valeur : playbook_vars)
2. Lancer le Playbook sur localhost et afficher la valeur de **prio**
3. Ajouter une section `vars_files` avec un fichier contenant la variable **prio** (valeur : playbook_vars_files)
4. Créer un fichier (fichier.yml) avec la variable **prio** (valeur : @file_var)
5. Lancer le Playbook sur localhost avec `-e @fichier.yml`
6. Lancer le Playbook sur localhost avec `-e prio=varline`
7. Lancer le Playbook sur localhost avec `-e prio=varline` et `-e @fichier.yml`

8. Créer un `group_vars` avec **prio** (valeur : group_vars)
9. Lancer le Playbook sur localhost avec `-e prio=varline` et `-e @fichier.yml`
10. Créer un `host_vars` avec **prio** (valeur : host_vars)
11. Lancer le Playbook sur localhost avec `-e prio=varline` et `-e @fichier.yml`
12. Ajouter un `vars_prompt:` au Playbook et relancer le.

Résumé

- La documentation avec l'outil ansible-doc
- L'installation de Apache et la manipulation de fichier en mode `ad hoc`
- La syntaxe du format YAML
- Les playbooks en transposant l'installation de Apache dans un fichier de configuration
- Vérification des playbooks avec ansible-lint
- Les Variables et leurs hiérarchies