Formatux

TP Ansible Niveau 2

TP du parcours

Table des matières

Module 1 : Les variables . . . 2
- Exercice 1.1 : Affichage d'une variable globale . . . 2
- Exercice 1.2 : Utilisation d'une variable locale . . . 3
- Exercice 1.3 : Utilisation d'une variable locale de type dictionnaire . . . 3
- Exercice 1.4 : Utilisation d'une variable locale de type dictionnaire à plusieurs dimensions . . . 4
- Exercice 1.5 : Surcharger une variable locale par la ligne de commande . . . 4
- Exercice 1.6 : Externaliser les variables . . . 5

Module 2 : La gestion des boucles . . . 6
- Exercice 2.1 : Créer des utilisateurs en boucle . . . 6
- Exercice 2.2 : Créer des utilisateurs en boucle (version avancée) . . . 6
- Exercice 2.3 : Afficher les valeurs d'un dictionnaire en boucle . . . 8

Module 3 : Les conditions . . . 9
- Exercice 3.1 : Installation de logiciel avec conditions . . . 9

Module 4 : La gestion des fichiers . . . 11
- Exercice 4.1 : Utiliser le module ini_file . . . 11
- Exercice 4.2 : Utiliser le module lineinfile . . . 12
- Exercice 4.3 : Utiliser le module copy . . . 12
- Exercice 4.4 : Utiliser le module fetch . . . 13
- Exercice 4.5 : Déployer un template . . . 14
- Exercice 4.6 : Utiliser le module get_url . . . 14

Module 5 : Les handlers . . . 16
- Exercice 5.1 : Gestion des handlers . . . 16

Module 6 : Les rôles . . . 17
- Exercice 6.1 : Créer un rôle . . . 17

Module 7 : Les tâches asynchrones . . . 19
- Exercice 7.1 : Attendre la fin d'une tâche . . . 19

🎓 Objectifs

- ☐ Utiliser les variables ;
- ☐ Mettre en oeuvre des boucles et des conditions ;
- ☐ Gérer les fichiers ;
- ☐ Envoyer des notifications et réagir ;
- ☐ Gérer les fichiers ;
- ☐ Créer des tâches asynchrones.

Module 1 : Les variables

Objectifs

- Utiliser les variables dans un playbook

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_variables.html

Exercice 1.1 : Affichage d'une variable globale

Ecrire un playbook `play-vars.yml` permettant d'afficher le nom de la distribution de la cible ainsi que sa version majeure, en utilisant des variables globales :

- Pour trouver la variable globale à utiliser :

```
$  ansible ansiblecli -m setup | grep "distribution"
        "ansible_distribution": "CentOS",
        "ansible_distribution_file_parsed": true,
        "ansible_distribution_file_path": "/etc/redhat-release",
        "ansible_distribution_file_variety": "RedHat",
        "ansible_distribution_major_version": "7",
        "ansible_distribution_release": "Core",
        "ansible_distribution_version": "7.4.1708",
```

Créer le fichier `play-vars.yml` avec ce contenu :

```
~
~
~
~
~
~
~
~
~
```

Executer le playbook :

```
~
~
~
```

Exercice 1.2 : Utilisation d'une variable locale

- Créer une variable locale nommée `port_http` contenant la valeur `8080`. Utiliser cette variable dans un message :

```
~
~
~
~
~
~
~
```

Exercice 1.3 : Utilisation d'une variable locale de type dictionnaire

- Créer une variable nommée `service` de type dictionnaire avec comme entrées :

```
service:
  name: apache
  rpm: httpd
```

- Créer une tâche permettant d'afficher les entrées du dictionnaire.

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

```
~
```

Exercice 1.4 : Utilisation d'une variable locale de type dictionnaire à plusieurs dimensions

- En vous inspirant de l'exercice précédent, créer un dictionnaire de dictionnaires. La variable `service` aura alors cette structure :

```
service:
  web:
    name: apache
    rpm: httpd
  db:
    name: mariadb
    rpm: mariadb-server
```

- Utiliser cette structure pour afficher selon une variable `type` les entrées du dictionnaire `web` ou `db`.

```
---
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Exercice 1.5 : Surcharger une variable locale par la ligne de commande

- Surcharger la variable `type` de l'exercice 1.4 en utilisant la ligne de commande :

```
~
~
```

Exercice 1.6 : Externaliser les variables

- Créer un fichier `vars.yml` contenant les variables définies précédement :

```
---
port_http: 8080
type: web
service:
  web:
    name: apache
    rpm: httpd
  db:
    name: mariadb
    rpm: mariadb-server
```

- Utiliser ce fichier de variables depuis le fichier `play-vars.yml`

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

- Tester le bon fonctionnement, avec et sans la surcharge de variable :

```
~
```

```
~
```

Module 2 : La gestion des boucles

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_loops.html

Exercice 2.1 : Créer des utilisateurs en boucle

- Créer un playbook `play-users.yml` qui créera 3 utilisateurs. Ces trois utilisateurs appartiennent au groupe `users`.

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Exercice 2.2 : Créer des utilisateurs en boucle (version avancée)

- Reprendre l'exercice précédent et l'améliorer.

Vous utiliserez un fichier de variables externe `users.yml` qui contiendra un dictionnaire d'utilisateurs. Chaque utilisateur aura comme propriété : un nom, un groupe, un état (present,absent).

Le fichier de variables users.yml

```
users:
  antoine:
    group: users
    state: present
  erwan:
    group: users
    state: present
  philippe:
    group: users
    state: absent
```

La boucle utilisée sera de type `with_dict: "{{ users }}"`.

- Le nom de l'utilisateur sera récupéré avec la variable `{{ item.key }}`.
- Le groupe sera récupéré avec la variable `{{ item.value.group }}`, l'état avec `{{ item.value.state }}`.

Le playbook play-users.yml

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Exercice 2.3 : Afficher les valeurs d'un dictionnaire en boucle

- Reprendre l'exercice 1.6 en vous inspirant de l'exercice précédent. Le playbook permettra d'afficher plusieurs services.

```
~
~
~
~
~
```

Module 3 : Les conditions

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_conditionals.html

Exercice 3.1 : Installation de logiciel avec conditions

En vous inspirant de l'exercice 1.6 :

- créer deux tâches, la première permettant d'installer le service web, la seconde le service de base de données
- en utilisant le dictionnaire `service`
- en fonction de la valeur de la variable `type` fournie en arguments à la ligne de commande
- uniquement sur les distribution CentOS

```
---
type: web
service:
  web:
    name: apache
    rpm: httpd
  db:
    name: mariadb
    rpm: mariadb-server
```

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

```
~
~
```

Module 4 : La gestion des fichiers

Plus d'informations sur http://docs.ansible.com/ansible/latest/list_of_files_modules.html

Exercice 4.1 : Utiliser le module ini_file

- Modifier le fichier `/etc/yum.conf` pour exclure le kernel des paquets mis à jour.

Pour cela :

- vous allez utiliser le module `ini_file` pour modifier le fichier de configuration de yum.
- le fichier `/etc/yum.conf` doit contenir :

```
[main]
...
exclude=kernel*
```

Il n'y a pas d'espace avant et après le '=' d'une option dans le fichier `/etc/yum.conf`. Vous allez donc devoir utiliser le paramètre `no_extra_spaces=True` du module `ini_file`.

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Exercice 4.2 : Utiliser le module lineinfile

- Créer un nouveau playbook qui permettra d'exclure également les paquets `java*` de la mise à jour avec le module `lineinfile`.

> Le module `ini_file` permet d'ajouter une option dans une section précise d'un fichier `ini` contrairement à `lineinfile` qui est moins précis de ce point de vue là.
>
> Maintenant que nous savons que le fichier contient la ligne `exclude=` nous pouvons utiliser le module `lineinfile` pour la modifier.

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Exercice 4.3 : Utiliser le module copy

Depuis le serveur Ansible :

- Créer un fichier local `testfile.txt`
- Créer un playbook `sendfile.yml` qui envoie ce fichier vers le client

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

- Modifier le fichier sur le serveur, envoyer le fichier à nouveau sur le client
- Modifier le fichier sur le client, envoyer le fichier à nouveau sur le client. Vérifier.

Exercice 4.4 : Utiliser le module fetch

- Créer un playbook `getfile.yml` qui copie sur le serveur le fichier `testfile.txt` de l'exercice précédent présent sur le client.

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Exercice 4.5 : Déployer un template

Les templates jinja feront l'objet d'un module plus complet sur la question.

- Sur le serveur Ansible, créer le modèle suivant :

```
Je suis un fichier de test
Je suis déployé sur le serveur {{ inventory_hostname  }}
```

- Créer un playbook sendtemplate.yml permettant de déployer ce template

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

- Vérifier sur le client

```
$ cat test.txt
Je suis un fichier de test
Je suis déployé sur le serveur 192.168.10.11
```

Exercice 4.6 : Utiliser le module get_url

- Créer un playbook geturl.yml permettant de télécharger sur le client le fichier https://ftp.drupal.org/files/projects/drupal-8.4.4.tar.gz

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Module 5 : Les handlers

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_intro.html#handlers-running-operations-on-change

Exercice 5.1 : Gestion des handlers

- Créer un playbook qui :
  - installe httpd,
  - démarre le service,
  - modifie la valeur du ServerAdmin dans `/etc/httpd/conf/httpd.conf`
  - redémarre le service si besoin

Lancer plusieurs fois votre playbook en changeant la valeur du ServerAdmin et vérifier que le Handler soit bien exécuté !

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

Module 6 : Les rôles

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_reuse_roles.html

Exercice 6.1 : Créer un rôle

- Créer un rôle en reprenant l'exercice précédent
- Initialiser le rôle :

```
~
```

- Définir les variables du rôle dans le fichier `./serveurweb/vars/main.yml` :

```
~
~
~
~
```

- Définir les handlers du rôle dans le fichier `./serveurweb/handlers/main.yml` :

```
~
~
~
~
~
~
```

- Définir les tâches qui composent le rôle dans le fichier `./serveurweb/tasks/main.yml` :

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```

- Définir notre playbook `deploy-httpd-role.yml` qui utilise le rôle :

```
~
~
~
~
~
~
```

- Exécuter le playbook

Module 7 : Les tâches asynchrones

Plus d'informations sur http://docs.ansible.com/ansible/latest/playbooks_async.html

Exercice 7.1 : Attendre la fin d'une tâche

- Créer un playbook qui met à jour le client, le redémarre sans afficher de message d'erreur.

Ici toute l'astuce est d'attendre 2s puis de lancer le reboot pour éviter que le serveur ne coupe la connexion avant la fin du playbook !

```
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
```