# 1. Les Variables

Les variables sont des données arbitraires pouvant être déclarées à divers endroits et qui peuvent être utilisée dans les « playbooks », « roles » et « templates » ansible.

Les variables ansible sont des constantes et leur valeur ne peut pas changer durant l'exécution des codes ansible.

Ansible dispose de différents types de variables tels que :

- Integers, Strings, Booleans, Dictionnaires, Listes/tableaux

## 1.1. Les variables classiques et dictionnaires

Dans cette partie vous allez utiliser des variables simples pour paramétrer :

- L'interface d'écoute du serveur MySQL (Via Variable classique de type « string »)

- L'utilisateur MySQL à créer. ( Via « Dictionnaire »)

### 1.1.1. Personnalisation interface d'écoute mysql – Variables classiques

Dans cette partie vous allez ajouter au playbook réalisé lors de l'atelier précédent la déclaration de la variable « mysql_bind_address ».

playbook4.yaml

```
---
- hosts: client2
  become: true
  vars:
    mysql_bind_address: adresse ip privé de la machine client2
  tasks:
  - name: Vérification de la connexion
    ping:
  - name: installation package serveur MySQL [mysql-server]
    apt:
      name: mysql-server
      state: latest
      update_cache: yes
  - name: Installation package python3-pymysql
    apt:
      name: python3-pymysql
      state: latest
  - name: Autorisation connexions à distance
    lineinfile:
      path: /etc/mysql/mysql.conf.d/mysqld.cnf
      regexp:   '^bind-address'
      line: "bind-address = {{ mysql_bind_address }}"
    notify:
      - Relancement du service mysql
  - name: Gestion du service mysql
    service:
      name: mysql
      state: started
      enabled: yes
```

```
  - name: Création de l utilisateur boby
    mysql_user:
      name: boby
      host: 10.10.20.4
      password: azerty
      state: present
      priv: '*.*:ALL'
      login_unix_socket: /var/run/mysqld/mysqld.sock

  handlers:
  - name: Relancement du service mysql
    service:
      name: mysql
      state: restarted
```

2) Lancez l'exécution du Playbook via la commande suivante :

ansible-playbook playbook4.yaml

3) Le Playbook est-il idempotent ?

4) Vérifiez dans le fichier de configuration du serveur MySQL si la variable à bien été remplacée par sa valeur.

Dans cette partie vous allez utiliser un dictionnaire « mysql_users » pour personnaliser l'utilisateur à créer dans sur le serveur MySQL.

Les dictionnaires sont très utiles lorsqu'il s'agit de déclarer un ensemble de variables pour une même entité.

1) Créez un fichier de playbook nommé « playbook5.yaml » avec le contenu suivant

```
---
- hosts: client2
  become: true
  vars:
    mysql_bind_address: IP_CLIENT_2
    mysql_users:
      pepito:
        priv: '*.*:ALL'
        host: 'IP_CLIENT_1'
        password: 'azerty'

  tasks:
  - name: Vérification de la connexion
    ping:

  - name: installation package serveur MySQL [mysql-server]
    apt:
      name: mysql-server
      state: latest
      update_cache: yes

  - name: Installation package python3-pymysql
    apt:
      name: python3-pymysql
      state: latest

  - name: Autorisation connexions à distance
    lineinfile:
      path: /etc/mysql/mysql.conf.d/mysqld.cnf
      regexp:   '^bind-address'
      line: "bind-address = {{ mysql_bind_address }}"
    notify:
      - Relancement du service mysql

  - name: Gestion du service mysql
    service:
      name: mysql
      state: started
      enabled: yes
  - name: Création de l utilisateur MySQL
    mysql_user:
      state: present
      name: "{{ item.key }}"
      host: "{{ item.value.host }}"
      password: "{{ item.value.password }}"
      priv: "{{ item.value.priv }}"
      login_unix_socket: /var/run/mysqld/mysqld.sock
    loop: "{{ query('dict', mysql_users) }}"
```

```
handlers:
- name: Relancement du service mysql
  service:
    name: mysql
    state: restarted
```

Note : Comme vous pouvez le constater, la tâche « Création de l'utilisateur MySQL » utilise une boucle « loop » pour parcourir tous les éléments du dictionnaire.

Attention : Actuellement, le dictionnaire ne dispose que d'une seule clé (« pepito »), nous devons donc utiliser la fonction « query('dict', dictionary_name) » pour récupérer tous les éléments de la clé.

Si plusieurs clés sont spécifiées, il faut utiliser la fonction « lookup('dict', dictionary_name) ».

2) Lancez l'exécution du playbook via la commande suivante :

ansible-playbook playbook5.yaml

3) Le playbook est-il idempotent ?

4) Valider la bonne création de l'utilisateur « pepito » en initiant une connexion à la base de données mysql depuis le client « client1 » avec l'utilisateur « pepito ».

5) Modifiez le playbook précédent pour ajouter la création d'un second utilisateur dans le dictionnaire « mysql_users », tout en conservant la déclaration de l'utilisateur « pepito ».

**se sera le playbook : playbook5_bis.yaml**

6) Lancez le déploiement de votre playbook. Est-il idempotent ?

7) Validez la bonne création de votre utilisateur en initiant une connexion à la base de données mysql depuis le client « client1 » avec votre utilisateur.

Dans cette partie vous allez utiliser un tableau pour installer une liste de packages en une seule tache.

L'objectif est de déclarer un tableau de packages « mysql_packages » à installer pour notre serveur MySQL (donc les packages « mysql-server » et « python-mysqldb ») et avoir juste une seule tache utilisant le module « apt » et à laquelle nous passerons le tableau de packages à installer.

1) Créez le playbook nommé « playbook6.yaml » avec le contenu suivant :

```
---
- hosts: client2
  become: true
  vars:
    mysql_bind_address: IP_CLIENT_2
    mysql_users:
      pepito:
        priv: '*.*:ALL'
        host: 'IP_CLIENT_1'
        password: 'azerty'
    mysql_packages:
    - mysql-server
    - python3-pymysql

  tasks:
  - name: Vérification de la connexion
    ping:
  - name: installation packages serveur MySQL [mysql-server, python3-pymysql]
    apt:
      name: "{{ mysql_packages }}"
      state: latest
      update_cache: yes

  - name: Autorisation connexions à distance
    lineinfile:
      path: /etc/mysql/mysql.conf.d/mysqld.cnf
      regexp:   '^bind-address'
      line: "bind-address = {{ mysql_bind_address }}"
    notify:
      - Relancement du service mysql

  - name: Gestion du service mysql
    service:
      name: mysql
      state: started
      enabled: yes

  - name: Création de l utilisateur boby
    mysql_user:
      state: present
      name: "{{ item.key }}"
      host: "{{ item.value.host }}"
      password: "{{ item.value.password }}"
      priv: "{{ item.value.priv }}"
      login_unix_socket: /var/run/mysqld/mysqld.sock
    loop: "{{ query('dict', mysql_users) }}"
```

```
handlers:
- name: Relancement du service mysql
  service:
    name: mysql
    state: restarted
```

Note : Comme vous pouvez le constater, aucune boucle n'est nécessaire pour la tâche « installation packages serveur MySQL [mysql-server,python-mysqldb] ».
Cela est dû au fait que nativement, l'argument « name » du module « apt » supporte les tableaux.
2) Lancez l'exécution du playbook.
3) Le playbook est-il idempotent ?

Dans cette partie vous allez créer une variable qui va indiquer quelle est la version de MySQL installée sur le serveur.

1) Ajoutez au playbook précédent le code suivant , appelez le playbook6-1.yaml:

```
- name: Récupération MySQL Version
  shell: mysql --version | awk '{print $5}' | awk 'gsub(",","")'
  register: mysql_version
  changed_when: False
- name: Affichage version MySQL
  debug:
    msg: "{{ mysql_version.stdout }}"
```

Comme vous pouvez le constater, la tâche « Récupération MySQL Version » dispose de nombreux paramètres.

Le module « shell » permet d'exécuter des scripts sur les machines distantes. Le script utilisé ici renvoi uniquement la version de MySQL.

Le paramètre « register » enregistre dans la variable « mysql_version » le retour de la tâche « Récupération MySQL Version »

Le paramètre « changed_when : False » permet d'indiquer que cette tâche ne fait pas de modification sur le système distant.

La tâche « Affichage version MySQL » affiche le contenu « stdout » de la variable « mysql_version ».

2) Lancez l'exécution de votre playbook.

3) Le playbook est-il idempotent ?

4) Quels sont les autres champs retournés par la variable « mysql_version » ?

Les Facts sont des variables contenant des informations découvertes sur les clients Ansible.

Les Facts sont très utiles pour personnaliser l'exécution des codes Ansible sur les clients en fonction de leur environnement.

Dans cette partie vous allez modifier le playbook précédent pour indiquer au serveur MySQL d'écouter sur l'interface principale de la machine.

1) Listez tous les Facts disponibles sur le client2 via la commande suivante :

ansible client2 -m setup

2) Identifiez le « fact » ansible retournant l'adresse IPv4 interne

a. Attention, vous devez récupérer le fact permettant de cibler la bonne interface réseau, à savoir « ens4 ». Même si l'IP se retrouve dans d'autres paramètres, il faut sélectionner le bon, pour la reproductibilité du playbook.

3) Créez le playbook « playbook7.yaml » qui modifie l'interface d'écoute du serveur MySQL pour que celui-ci écoute sur le « fact » identifié précédemment. (Reprenez le code du playbook précédent.)

4) Lancez votre playbook et vérifiez que le fichier de configuration du serveur MySQL est bien modifié correctement.