**Compte-Rendu**

**Cours Ansible**

Préparer par : din15

Formateur : abd errahman

Fait le : 29/03/2023

# Introduction :

Ansible est une solution Open-source, développée en langage Python par Michael DeHaan en 2012. Ansible est une plate-forme logicielle libre pour la configuration et la gestion des ordinateurs. Elle combine le déploiement de logiciels multi-nœuds, l'exécution des tâches ad-hoc, et la gestion de configuration.

Ansible ne nécessite pas de socle « client » pour les machines à déployer : il se base sur le protocole SSH et un interpréteur Python. Pas besoin de serveur imposant pour l'exploiter, car un simple ordinateur Linux peut suffire.

Red Hat rachète Ansible Inc. en octobre 2015. La documentation de Ansible se situe ici : <https://docs.ansible.com/>

Vous pouvez utiliser Ansible pour automatiser trois types de tâches :

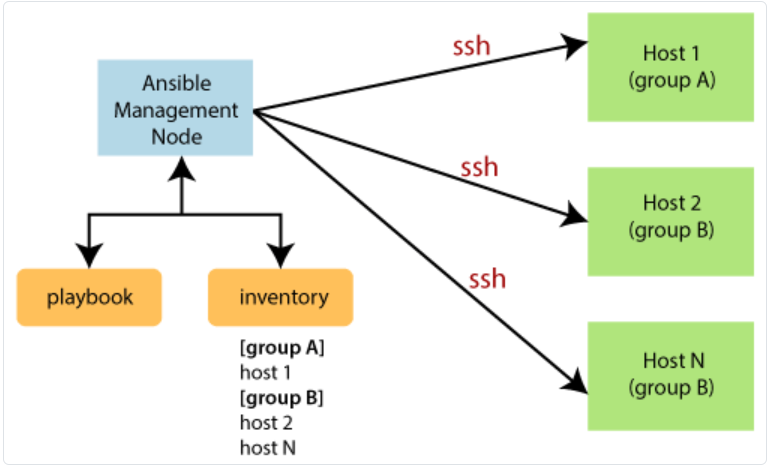
* **Provisioning** : configurer les différents serveurs dont vous avez besoin dans votre infrastructure
* **Configuration** : modifier la configuration d'une application, d'un système d'exploitation ou d'un périphérique, démarrer et arrêter les services, installer ou mettre à jour des applications, mettre en œuvre une politique de sécurité, ou effectuer une grande variété d'autres tâches de configuration.
* **Déploiement d'applications** : adopter une démarche DevOps en automatisant le déploiement d'applications développées en interne sur vos environnements de production.

Comment fonctionne Ansible :

Dans Ansible, il existe deux catégories d'ordinateurs: le **nœud maître** (master) et les **nœuds esclaves** (slaves). Le nœud maître est une machine sur laquelle est installé l'outil Ansible. Il doit y avoir au moins un nœud maître, bien qu'un nœud maître de sauvegarde puisse également exister.

Ansible fonctionne en se connectant à vos nœuds en **SSH** et en y poussant de petits programmes, appelés **modules**. Ces modules sont définis dans un fichier nommé le **Playbook**. Le nœud maître, se base sur un fichier d'inventaire qui fournit la liste des hôtes sur lesquels les modules Ansible doivent être exécutés.

Ansible exécute ces modules en SSH et les supprime une fois terminé. La seule condition requise pour cette interaction est que votre nœud maître Ansible dispose d'un accès de connexion aux nœuds esclaves. Les clés SSH sont le moyen le plus courant de fournir un accès, mais d'autres formes d'authentification sont également prises en charge.

Voici un schéma qui reprend notre explication :

# Installation et Déploiement de Ansible :

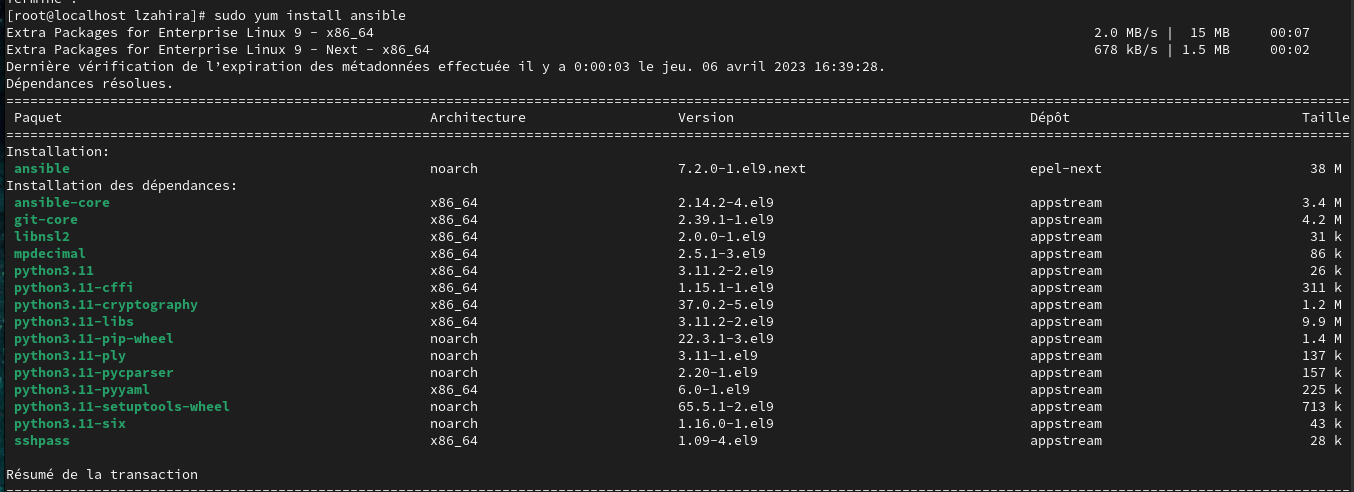
### Installation ansible nœud master :

Il existe **différentes façons d'installer le moteur Ansible**. Vous avez le choix entre les méthodes suivantes :

* Installation de la dernière version proposée par votre **gestionnaire de package** de votre OS.
* Installation avec le **gestionnaire de packages python pip**.
* Installation depuis les sources Ansible afin d'utiliser et tester les dernières fonctionnalités.

L'installation se tient sur une ligne de code :

* # yum install epel-release
* # yum install ansible

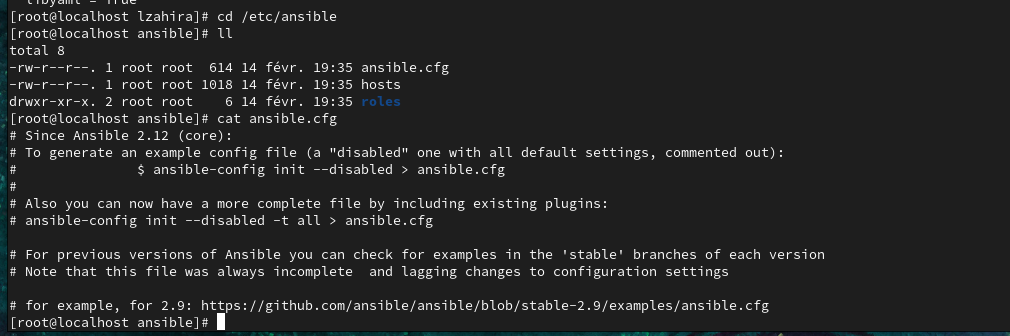


### Configuration du Master Node ansible :

Le fichier **/etc/ansible/ansible.cfg** est le fichier de configuration global d'Ansible qui peut être utilisé pour définir les paramètres par défaut pour toutes les tâches Ansible exécutées sur une machine. Les paramètres les plus importants dans ce fichier sont :

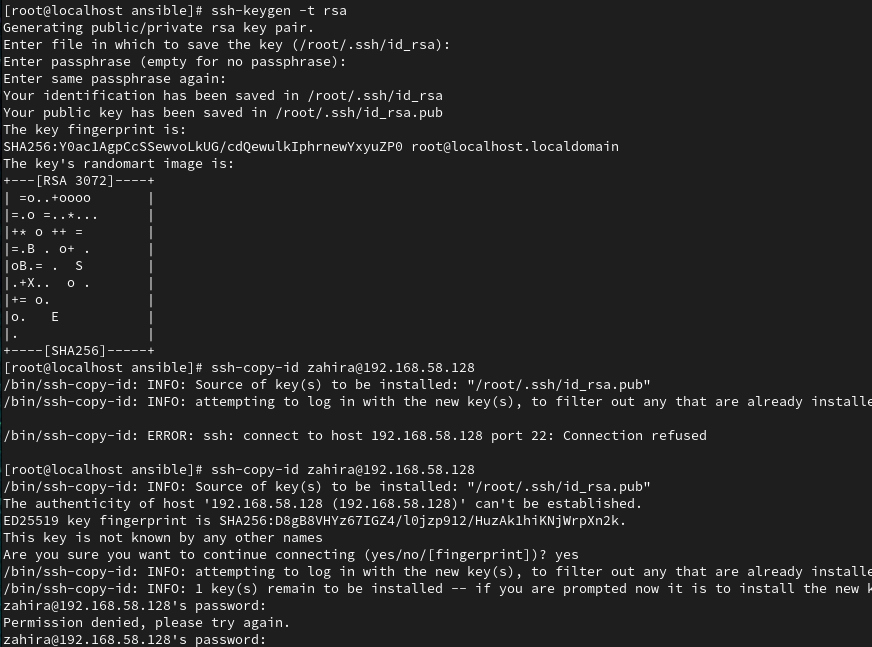
* **inventory**: Le chemin vers le fichier d'inventaire par défaut qui contient une liste des hôtes sur lesquels les tâches Ansible seront exécutées.
* **remote\_user** : Le nom d'utilisateur à utiliser pour se connecter aux hôtes distants. Si aucun nom d'utilisateur n'est spécifié pour une tâche, Ansible utilisera la valeur définie dans ce paramètre.
* **become** : Un booléen qui indique si Ansible doit exécuter les commandes en tant qu'utilisateur root sur l'hôte distant. Si la valeur est "yes", Ansible exécutera les commandes en tant que superutilisateur.
* **become\_method** Le mode de passage à l'utilisateur root sur l'hôte distant. Les options courantes sont "sudo" et "su".
* **become\_user :**Le nom d'utilisateur à utiliser lors de la transition vers l'utilisateur root sur l'hôte distant.
* **module\_name\_patters** : Une liste de motifs de noms de modules pour lesquels Ansible doit chercher des plugins. Les plugins sont des extensions qui ajoutent des fonctionnalités à Ansible.
* **retry\_files\_enabled**: Un booléen qui indique si Ansible doit stocker les fichiers de reprise. Ces fichiers peuvent être utilisés pour reprendre les tâches interrompues en cas d'échec ou de coupure de connexion.
* **fact\_caching**: Un booléen qui indique si Ansible doit conserver en cache les informations sur les hôtes distants. Cette option peut accélérer les tâches répétitives en évitant de récupérer les mêmes informations à chaque exécution.
* **log\_path** :Le chemin vers le fichier de journalisation d'Ansible, qui contient les informations sur les tâches exécutées par Ansible.

Il est important de noter que la signification et l'utilisation de chaque paramètre peuvent varier en fonction de la configuration de votre environnement et des tâches que vous exécutez avec Ansible.

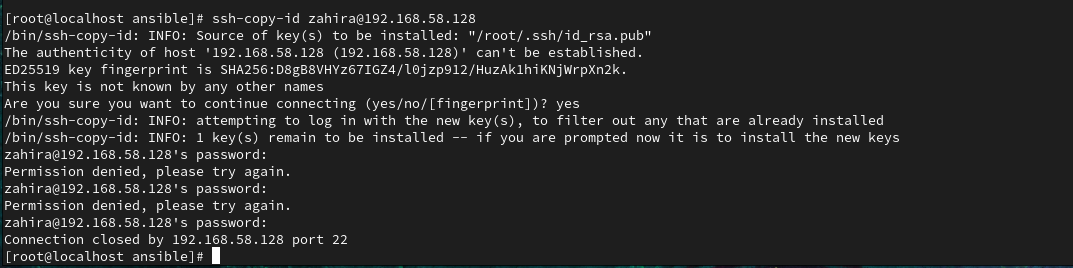


### Configuration de la connexion Shell Secure :

Ansible communique avec des machines distantes via le protocole SSH. Il faut donc vous assurez que vous pouvez vous connecter à vos différents nœuds distants de votre inventaire en utilisant le protocole SSH. Dans notre cas nous utiliserons la commande ssh-copy-id afin d'ajouter la clé SSH publique du serveur de contrôle au fichier *authorized\_keys* sur nos systèmes gérés.



Ajout de clé ssh pour le premier serveur ‘’’zahira’’ sur CentOS :

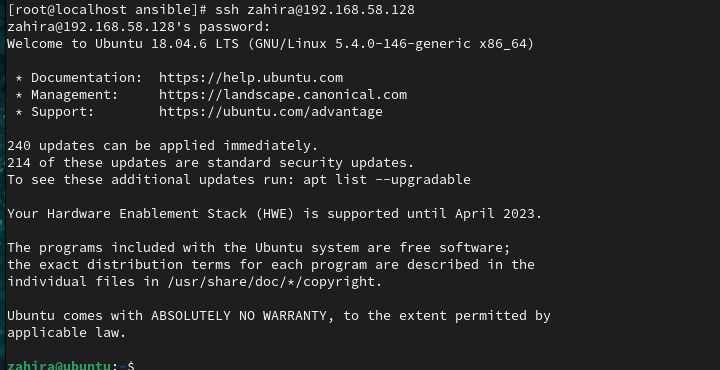


Si on a plusieurs serveurs qu’on veut déployer on utilise la solution du linux ‘**’expect’’**

### Configuration de nœud client sur linux :

On utilise la commande : **ssh user@ip**

* serveur ‘’’zahira’’ sur CentOS :



# Inventory Ansible :

### Introduction :

Ansible lit les informations sur les machines que vous souhaitez gérer à partir d'un fichier nommé **l'inventaire**. Bien que vous puissiez transmettre une adresse IP en paramètre avec la commande ansible, vous aurez besoin d'un inventaire pour profiter de la flexibilité et de la répétabilité complètes d'Ansible.

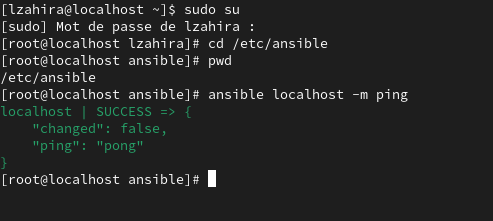
Vous pouvez définir vos machines esclaves dans le fichier **inventaire** qui est situé par défaut dans le fichier **/etc/ansible/host.** Votre inventaire peut stocker les adresses IP ou les noms de domaines complets de vos machines distantes (mais aussi des groupes, des alias ou des variables).

### Commandes de base sur ansible :

**D'abord qu'est-ce qu'un module ?**

Les modules sont comme de petits programmes qu'Ansible pousse depuis une machine de contrôle vers tous les hôtes distants. Les modules sont exécutés à l'aide de **playbooks** ou depuis la cli Ansible, et ils contrôlent des éléments tels que les services, les packages, les fichiers et débogage et plus.

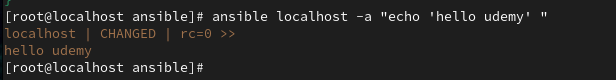
Dans notre exemple, on utilisera le module ping . Ouvrez donc un terminal depuis votre machine de contrôle et lancez-y la commande suivante : ansible localhost -m ping



Voici une liste d'explication des différentes options de notre commande :

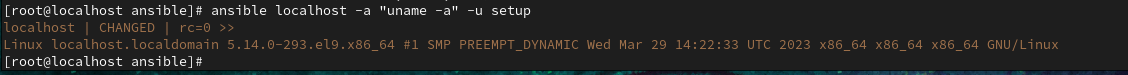
* all : ici on demande à Ansible d'exécuter la commande sur tous les hôtes de notre inventaire
* -m ping : ici on lui demande d'utiliser le module [ping](https://docs.ansible.com/ansible/latest/modules/ping_module.html)
* -u <user> : ici on lui demande de lancer notre module depuis l'utilisateur ‘’user ‘’

Vous pouvez aussi **exécuter directement des commandes Linux sur Ansible**, en utilisant tout simplement l'option **-a**. Par exemple :

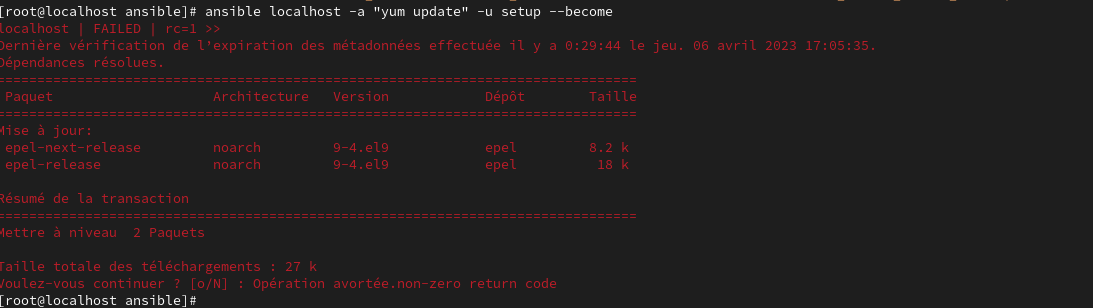


Exécuter une commande pour récupérer les informations de notre machine :

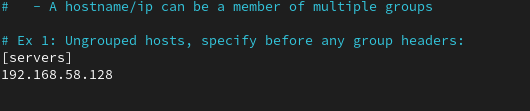
ansible localhost -a ‘’ uname -a ‘’ -u setup



si on veut installer un élément par exemple « yum update » et il faut être en root donc on peut utiliser - - become pour passer automatiquement à root pour que la commande s’exécute :



Le fichier inventaire *"* ***hosts*** *"* se trouve dans ***"/etc/ansible*** *"* par défaut. Il définit la liste des machines   
qui pourrait répondre aux requêtes d’Ansible. Les ordinateurs définis dans le fichier *" hosts "*peuvent être organisés en groupe. On peut même créer des groupes de groupe. Ce fichier est modifiable dans la configuration d’ansible ou même spécifiable à l’execution des comma,des ansible.



Dans notre exemple on souhaite créer un fichier vide nommé *test.txt* dans le dossier ***/home/docker/***et ne lui fournir que des droits de lecture (0644).

* **path** (obligatoire) : pour l'emplacement et le nom de notre fichier (chemin absolu + le nom de notre fichier)
* **state** : nous utiliserons la valeur **touch** pour obliger la création de notre fichier. message personnalisé et le paramètre
* **mode** : pour les autorisations que le fichier doit avoir dans notre cas ça sera le droit 0644

Pour cela, nous utiliserons l'option **-a** qui permet également de fournir des arguments complémentaires à l'option **-m** (module). Nous aurons ainsi la commande suivante :

$ ansible all -m file -a "path=/home/docker/test.txt state=touch mode=0644"

# Introduction au YAML :

La syntaxe YAML (Yet Another Markup Language) C’est un langage facile à lire et à écrire. On peut y   
inclure une syntaxe JSON. On peut y manipuler du texte, des listes et des dictionnaires. L’indentation est   
très strict (2 espaces / indentation). On démarre un playbook toujours avec --- puis on va à la ligne. Enfin   
les commentaires sont défini grâce au #

**Exemple :**

**---***# Liste des commutateurs juniper présent à l’IBMP*commutateurs**:  
-** ex3300  
**-** ex4200  
**-** ex2200  
**-** ex4300  
**...**

# Travailler avec les playbooks :

### Introduction aux playbooks :

Les playbooks permettent d’exécuter un ensemble de tâches à effectuer sur les machines hôtes de manière séquentielle sur tout ou partie de l’inventaire.

• hosts: file\_slow  
 tasks:   
 - name: Install apache  
 yum:  
 name: httpd  
 state: present  
 tags:  
 -install\_httpd

Chaque tâche utilise un module de Ansible. On utilise le langage YAML pour écrire un playbook. Un playbook est constitué au minimum :  
 • D’une variable *hosts* qui désigne les machines cibles  
 • D’une variable *task* qui définit une action à accomplir.  
Les playbooks peuvent aussi :  
 • Emettre des notifications, qui seront utiles pour déclencher une action dans certaines conditions  
 • Effectuer des actions conditionnelles selon leur valeur (par exemple le type de distribution du système)  
 • Utiliser des templates pour créer ou modifier des fichiers  
On trouve aussi dans les playbooks :  
 • Des rôles à utiliser  
 • Des handlers (tâches spécifique se déclenchant grâce à une notification)  
 • Des éléments de configuration pour Ansible (par exemple l’utilisation de sudo)  
 • Des variables spécifiques qui ne sont pas relié à l’inventaire.  
**Exemple :**

L’exécution du playbook install\_apache.yml se fait grâce à la commande ansible-playbook :  
 **$ ansible-playbook install\_apache.yml**  
Par défaut, un playbook s’exécute sur tous les hôtes concernés. Il est possible de de cibler ces actions grâce aux options :  
 • Tags : cette option se définie dans le playbook et on l’appelle **--tags nom\_du\_tag** ou **–skiptags**• L’option **–l** qui permet de cibler une machine en particulier  
**Exemple :**  
 **$ansible-playbook install\_apache.yml --tags** **install\_httpd -l nlame1.example.com**

### Développer un playbook :

Une image contenant texte

Description générée automatiquementOn va éditer le fichier inventaire **hosts** et on ajoute la machine cible pour notre exemple c’est la machine debian **amal :**

On va créer un fichier **setup\_apache.yml** dans le répertoire **playbooks** qu’on a déjà créer dans **/etc/ansible/**  :

Une image contenant texte

Description générée automatiquement

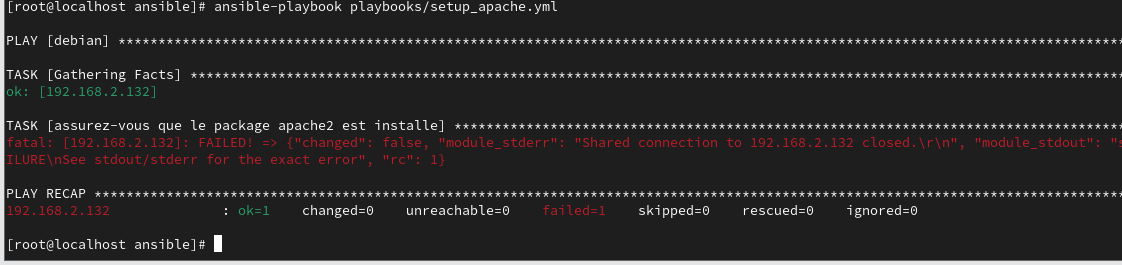
Dans le fichier **setup\_apache.yml** , définissez les tâches à effectuer en utilisant la syntaxe Ansible. Voici un exemple de configuration de base qui installe le paquet Apache :

Une image contenant texte

Description générée automatiquement

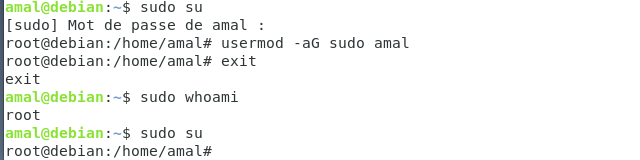
Enregistrez le fichier **setup\_apache.yml** et quittez l'éditeur de texte.

Exécutez le playbook Ansible en util0isant la commande suivante :

**$ ansible-playbook playbooks/setup\_apache.yml**

On ajoute l’utilisateur amal au groupe pour régler le problème avec la commande suivante pour Debian

**$ usermod -aG sudo amal**

****

Après on tape la commande suivante :

# ansible-playbook playbooks/setup\_apache.yml --user=amal --extra-vars "ansible\_sudo\_pass=pwd"

# Travailler avec les modules:

### Introduction aux modules insible :

Il s’agit d’une liste de fonctions et d’actions prédéfinies à appeler lors de l’utilisation d’Ansible, soit via des commandes Ado ou en utilisant des playbooks. Pour voir les modules on exécute la commande ansible-doc –l .

### Développer avec un module :

Une image contenant texte

Description générée automatiquementOn utilise le module copy dans un playbook :

Une image contenant texte

Description générée automatiquementEnsuite on tape la commande :

# ansible-playbook copy.yml --user=amal --extra-vars "ansible\_sudo\_pass=0605659259vde"

On vérifie l’existence du fichier sur la machine Ubuntu 18 et Debian 14 :

Une image contenant texte

Description générée automatiquement

# Une image contenant texte Description générée automatiquementtravailler avec les variables :

### Introduction aux variables ansible :

Les variables sont utilisées pour stocker des valeurs qui seront utilisées ultérieurement dans un playbook. Elles peuvent être définies et remplacées de plusieurs manières. Les machines peuvent être récupérées en tant que variables et être utilisées.

La définition des variables se fait soit en passant par un fichier de variables, soit en les déclarant dans le playbook.

Les prompts de variables sont des variables définies en lancement, par exemple, on met un prompt pour dire sur quel OS (si on plusieurs types de machines à gérer) on souhaite lancer une commande, on choisit par exemple tous les OS Ubuntu.

### Développer un playbook avec variable :

**Nomenclature des variables :**

Les noms donnés au variables sont présentés sous trois formes :

* **variable**
* **nom\_variable**
* **nom\_de\_variable**

Les noms chiffrés et les mots séparés par des espaces, des points ou tiret (-) ne sont pas pris en compte.

1. **Citer les variables :**

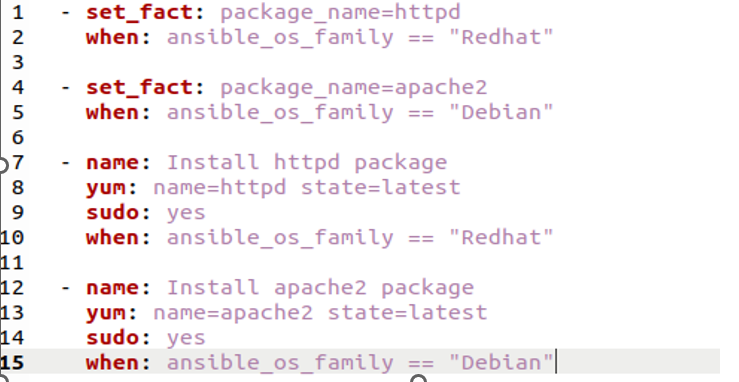
Par exemple on veut stocker une chaine de caractères dans une variable ensuite afficher sa valeur dans un message. Pour cela on crée le playbook suivant :

Une image contenant texte

Description générée automatiquementUne image contenant texte

Description générée automatiquement

### Développer des variables dans un fichier de taches inclues :

On peut stocker les variables dans un playbook, et ensuite l’importer dans un autre playbook.

# Travailler avec des conditionnalités :

On prend l’exemple précédent, on veut installer le serveur apache sur les machines (os redhat et debian). Donc l’installation se fera selon la syntaxe suivante :

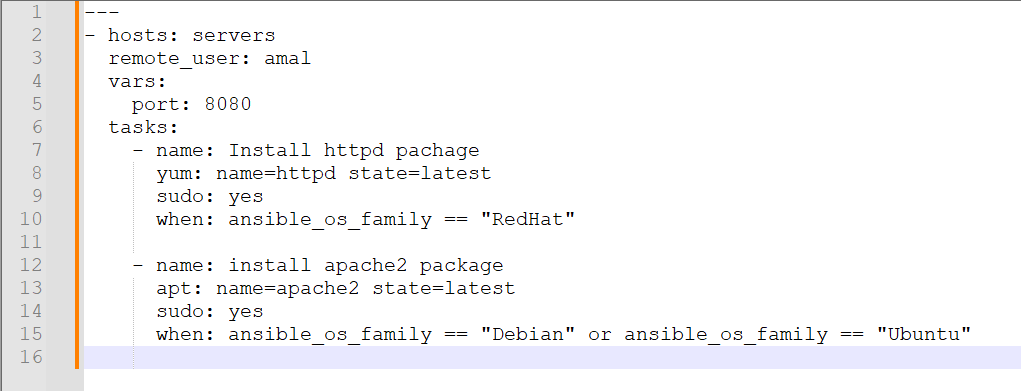
**if os = ‘’redhat system’’**

**install httpd**

**else if os = ‘’debian system’’**

**install apache2**

**end**

Pour ce faire on crée le playbook suivant :

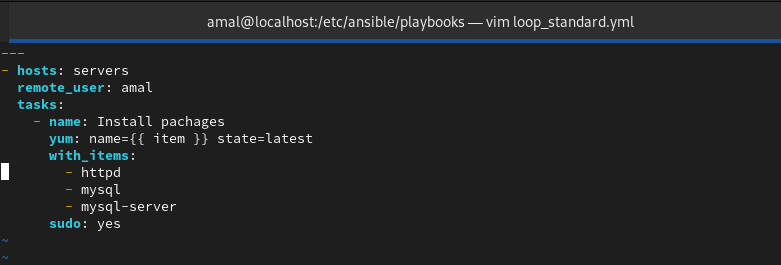
On utilise **when** pour intégrer les conditions.

# Travailler avec les boucles :

Sous Ansible on trouve trois types de boucles :

* **Les boucles standards** : permet de passer une liste de packages à installer, Ansible exécutera les tâches pour tous les packages répertoriés.

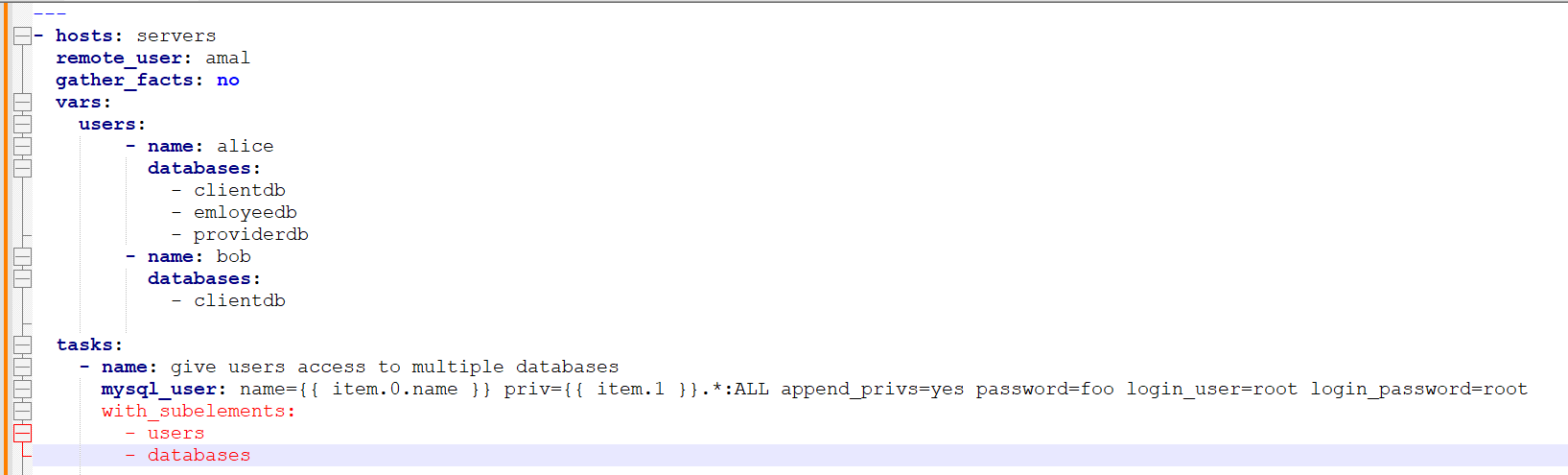
**Exemple :** ce playbook demande à Ansible d’exécuter l’installation du package plusieurs fois.



* **Les boucles imbriquées** : sont utiles lorsqu’on veut utiliser plusieurs opérations sur la même ressource. Par exemple, on veut autoriser l’accès à plusieurs bases de données à un ou à plusieurs utilisateurs de MySQL.

Une image contenant texte

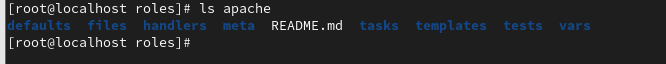
Description générée automatiquement

* **Les boucles sur les sous-éléments** : on utilise ce type de boucles, si on veut spécifier des ensembles de bases de données aux différents utilisateurs (boucler sur les sous éléments).

# Travailler avec les Roles :

Une image contenant texte

Description générée automatiquementLes rôles Ansible sont des fonctionnalités robustes d’Ansible qui facilitent la réutilisation et favorisent d’avantage la modularisation de la configuration. Ces rôles sont simples à configurer et permettent une complexité correcte. La structure des rôles est assez simple, il s’agit d’un groupe de variables (tâches, fichiers, et de gestionnaires) qui sont stockés dans une structure de fichiers normalisée.la partie importante d’un rôle consiste à rappeler la structure du répertoire. La commande utilisée dans la création des rôles est **ansible-galaxy init ‘NOM\_DU\_ROLE’**



Une image contenant texte

Description générée automatiquementOn tape la commande **tree apache/** pour voir la structure de ce rôle :

* Le répertoire **tasks** contiendra la liste principale des tâches à exécuter par le rôle.
* Le répertoire **handlers** contiendra des gestionnaires qui peuvent être utilisés par ce rôle ou d’autres rôle éventuellement.
* Le répertoire **defaults** contient des variables par défaut.
* Le répertoire **vars** contient les autres variables pour le rôle à la priorité la plus élevée par défaut.
* Le répertoire **files** contient les fichiers nécessaires au transfert ou déploiement sur des machines cibles via ce rôle.
* Le répertoire **templates** contient des modèles qui peuvent être déployés via ce rôle.
* Le répertoire **meta** définit certaines données et informations sur ce rôle.

Par exemple on a un playbook qui permet de déployer le serveur httpd, ce qui va changer c’est seulement l’host, donc on crée un rôle.

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquementPour ajouter un rôle on se déplace dans le répertoire ‘roles’, ensuite on se déplace dans le répertoire ‘tasks’, on modifie le fichier **main.yml** : on installe httpd, puis on copie le fichier de configuration et le fichier index.html, enfin on restart le serveur apache.

Une image contenant texte

Description générée automatiquement

On peut, au lieu de citer la tâche de configuration, importer des tâches. Donc on modifie le fichier main.yml et on met trois fichiers qu’on a créé ailleurs.

Une image contenant calendrier

Description générée automatiquementUne image contenant texte

Description générée automatiquementOn a les trois fichiers créés dans le répertoire tasks.

Une image contenant texte

Description générée automatiquementEnsuite on crée les deux fichiers qu’on a mentionnés dans le fichier configure.yml, dans le répertoire **/roles/apache/files**.

Une image contenant texte

Description générée automatiquementEnsuite on travaille sur la gestion, en se déplaçant dans le répertoire ‘handlers’ et on modifie le fichier main.yml pour redémarrer en cas de changement.

Puis on se déplace dans le répertoire meta, et on modifie le fichier main.yml en ajoutant des informations sur le rôle (la licence de la plate-forme prise en charge).

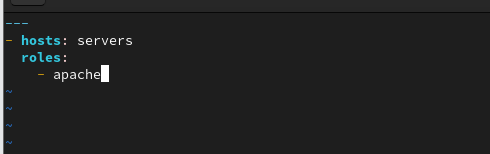
Une image contenant texte

Description générée automatiquement

On tape **tree** dans le répertoire **apache**. On trouve notre configuration mise en place.

Une image contenant texte

Description générée automatiquement

Ensuite on crée un setup qui sera exécuté et appellera les rôles en tapant la commande **vim runsetup.yml** (dans le répertoire personnel) en estimant que seulement les machines qui vont être changés pour un même rôle (apache).

Si on veut vérifier si la syntaxe est bonne, on tape la commande

**ansible-playbook /etc/ansible/runsetup.yml --syntax-check**

Ensuite on déploie ce playbook sur les machines :

**ansible-playbook –i .yml runsetup.yml**

On peut exécuter les commandes de shell dans Ansible en tapant la commande

**ansible –a ‘’commande\_shell’’ localhsot**

