**Lab 2– Playbooks**

1. **Premier Playbook :** 
   1. Ecrire votre premier playbook avec un seul play s’exécutant sur le groupe **précédents**. **(OK)**
   2. Désactiver la collecte automatique des facts. **(OK)**
   3. Ecrire une première tâche qui vous affiche la liste des paquets à mettre à jour et la tester. **(OK)**
   4. Ajouter une tâche qui vérifie la connexion aux hôtes. **(OK)**
   5. Ajouter une tâche de collecte des facts et une deuxième pour afficher les informations collectées. **(OK)**
   6. Installer le paquet **httpd**. Si l’installation réussit, démarrer le service **httpd** correspondant. **(OK)**

> touch playbook.yaml

> ansible-playbook playbook.yaml

**-** name**:** update db servers

hosts**:** precedants

remote\_user**:** root

tasks**:**

**-** name**:** ensure postgresql is at the latest version

yum**:**

name**:** postgresql

state**:** latest

**-** name**:** ensure that postgresql is started

service**:**

name**:** postgresql

state**:** started

---

- hosts: localhost

become: yes

gather\_facts: no

tasks:

- name: check if hosts are accessible

wait\_for:

host: '{{ (ansible\_ssh\_host|default(ansible\_host))|default(inventory\_hostname) }}'

port: 22

state: started

delay: 0

timeout: 1

- name: Gather the package facts

package\_facts:

manager: auto

- name: Print the package facts

debug:

var: ansible\_facts.packages

- name: Print all available facts

ansible.builtin.debug:

var: ansible\_facts

- name: install Apache server

yum:

name: httpd

state: latest

- name: enable and start Apache server

service:

name: httpd

enabled: yes

state: started

1. **Structures de contrôle :**
   1. Ecrire un playbook avec des structures conditionnelles qui effectuent les tâches suivantes (tester chaque tâche avant de passer à la suivante):
      1. Il affiche le message « nom\_hote utilise la distribution redhat » (resp. debian) si la distribution de la machine distante est Redhat ou Centos (resp. Debian ou Ubuntu) ; le nom\_hôte est le nom de la machine distante récupéré dans le playbook.

- name: Retrieve hostname

hosts: localhost

tasks:

- name: Retrieve the hostname

command: hostname

register: result

- debug:

var: hostname

* + 1. Une deuxième tâche qui affiche le message « nom\_hote utilise la distribution redhat 7» si la distribution utilisée est Redhat ou   
       Centos et la version (**ansibl\_distribution\_major\_version**) est 7.

- hosts: localhost

gather\_facts: yes

become: false

tasks:

- name: Distribution

debug: msg="{{ ansible\_distribution }}"

- name: Distribution version

debug: msg="{{ ansible\_distribution\_version}}"

- name: Distribution major version

debug: msg="{{ ansible\_distribution\_major\_version }}"

* + 1. Ajouter une tâche qui demande à l’utilisateur son nom et son domaine d’expertise ; s’il saisit « **devops** » comme expertise, on lui affiche le message « bienvenue nom » sinon « bye nom ».

- hosts: localhost

vars\_prompt:

- name: username

prompt: What is your username?

private: no

- name: expertise

prompt: What is your expertise domain?

private: no

tasks:

- name: Retrieve the hostname

command: hostname

register: result

- debug:

var: hostname

- name: Print a message

ansible.builtin.debug:

msg: 'Logging in as {{ username }}'

when: expertise == 'devops'

- debug:

msg: 'bye {{ username }}'

when: expertise != 'devops'

* 1. Ecrire un playbook avec des boucles **loop**:
     1. Ecrire une tâche qui installe les paquets suivants en une boucle :

- python

- python-setuptools

- python-dev

- build-essential

- python-pip

- python-mysqldb

- hosts: localhost

user: ansible

become: True

tasks:

- name: Install all the packages

apt:

name: '{{ item }}'

state: present

update\_cache: True

with\_items:

- python

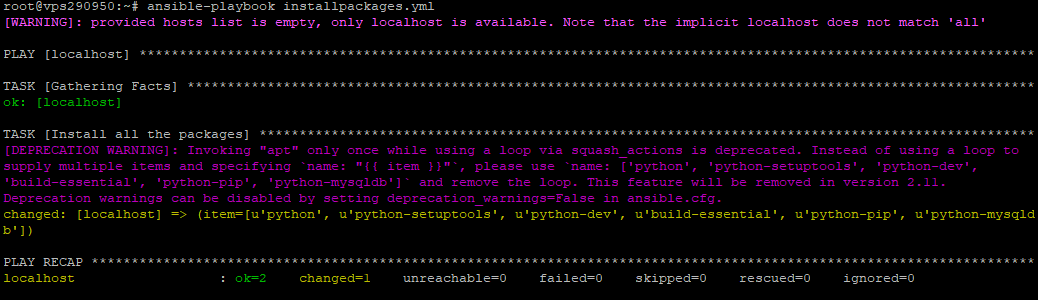
- python-setuptools

- python-dev

- build-essential

- python-pip

- python-mysqldb



Best solution:

- hosts: all

user: ansible

become: True

tasks:

- name: Install all the packages

apt:

name: "{{ item }}"

state: present

update\_cache: True

loop:

- python

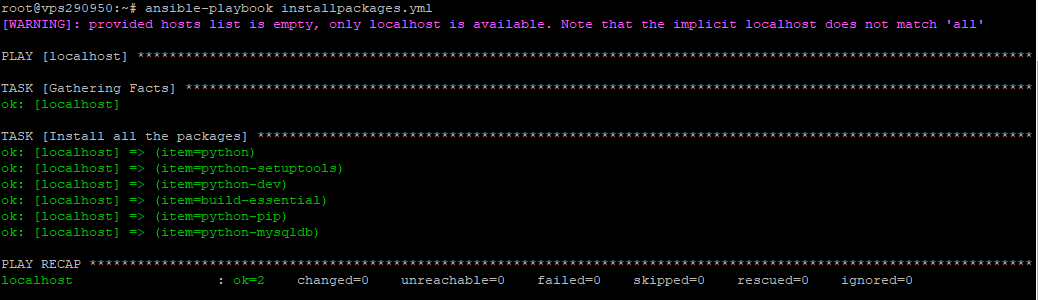
- python-setuptools

- python-dev

- build-essential

- python-pip

- python-mysqldb



**>ansible-playbook installpackages.yml -i inventory -l suivants**

* + 1. Ajouter 3 groupes : orsys, alten, ansible

* + 1. Ajouter 3 utilisateurs (user1, user2, user3) avec groupes secondaires orsys et alten et avec des mots de passe respectifs ansible, orsys et alten.

- hosts: localhost

user: ansible

become: True

tasks:

- name: Install all the packages

apt:

name: "{{ item }}"

state: present

update\_cache: True

loop:

- python

- python-setuptools

- python-dev

- build-essential

- python-pip

- python-mysqldb

- name: Ensure group "orsys" exists

group:

name: orsys

state: present

- name: Ensure group "alten" exists

group:

name: alten

state: present

- name: Ensure group "ansible" exists

group:

name: ansible

state: present

- name: make a new user "user1"

user:

name: user1

state: present

groups: "orsys, alten"

password: $6$c/CuRmitykVLHst$2lhp0ohRB8m/NDfGDBHWJyVlCGj5bKJD4ECjOMl90KPdvqKlqckkt901beDpElqVkTF8lBeA564j9I9Y1dsQw1

- name: make a new user "user2"

user:

name: user2

state: present

groups: "orsys, alten"

password: $6$EsFGXVrOT.7d4VJ$uVlT/Aw.5rkRPGc9//Vi3Zvi4M90fA3eyG8mbYfBKxH/5GFNjgmGp810Rnq2Kj9WnsYdYgG9NzOZFj6vwoY5m.

- name: make a new user "user3"

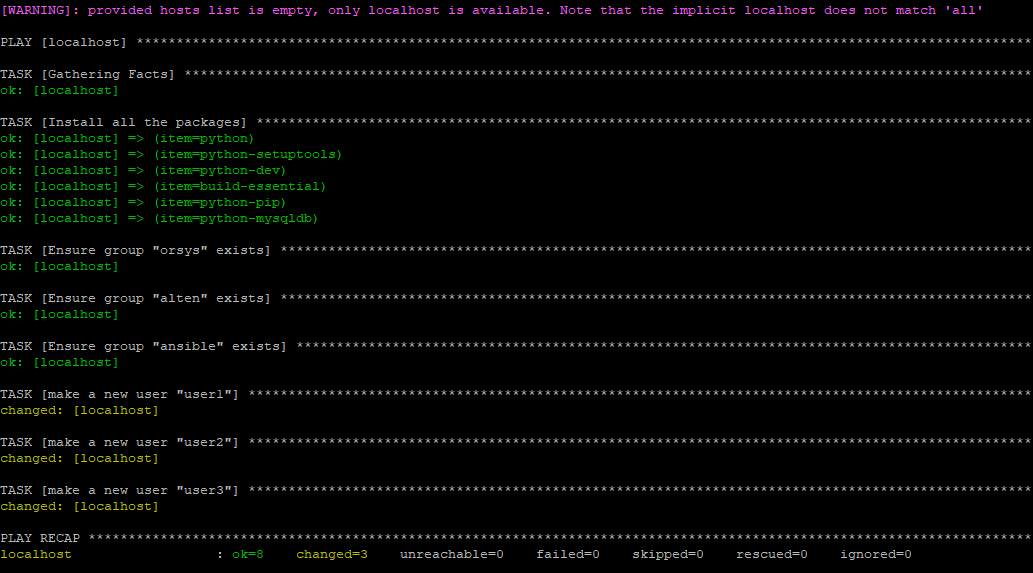
user:

name: user3

state: present

groups: "orsys, alten"

password: $6$OJSfPQ4aTmWKaW$FxP5o4A2lRYJSDX7WBvIctBBsT5Feua0harI82Qt7v88jWgYSEx2Vhnn36cIZG7bZmYfzcdGeYbz4jEN5BG6w1



* 1. On va maintenant utiliser le menu **vars\_files**:
     1. Créer un fichier appelé **packages** contenant la liste des paquets.

>touch packages.yml

packages\_to\_install:

- python

- python-setuptools

- python-dev

- build-essential

- python-pip

- python-mysqldb

* + 1. Créer un fichier appelé **groupes** contenant la liste des groupes g1, g2 et g3.

[usera@lx-14-100 ~]$ cat groups.yml

groupes:

- g1

- g2

- g3

* + 1. Créer un fichier appelé **users** contenant la liste des utilisateurs orsys, alten et (votre nom).

>touch users.yml

---

user\_passwords:

orsys: $6$rounds=656000$m/qpgaPV9nDhZA84$0Uz2fQ7PjnX.eMIDSlw0hUetHYat.VuxIzBNsbceZjg60XMe.0hrDekRybNAMe0fPqvczikY0Hdph8KMhcHct. # ws#P)Bg)l853

alten: $6$rounds=656000$RhhaEkZK/60KAYDf$U/nsycrW2A4SAuhBbAW4na4OLunPrUfR31OU3ThY1ge3vc.RUfhyHTg5dShkTYFGB/455lv0vOWDAmbGiOI730 # qbbw8&OeZ1ql

ahmed: $6$rounds=656000$aXLv86ermeammjFO$MooGjguTxUjhc2m6OefDddz0mszG/SprKiyTsND0lpT3f4.R7V5KucdK9JdLluOF.WnpGAz/GKy2umf5TPkPr. # zIPjxwCFm@ES

* + 1. Réécrire une deuxième version du playbook en utilisant la balise **vars\_files :** pour installer les paquets, ajouter les utilisateurs et groupes.

> ansible-playbook var\_packages.yml --extra-vars "ansible\_sudo\_pass=usera"

- hosts: localhost

user: ansible

become: True

tasks:

- name: Include variables

include\_vars: 'packages.yml'

- name: "Get installed packages"

yum:

list: "installed"

register: installed\_packages

- name: "Install missing packages"

package:

state: "present"

name: "{{ item }}"

with\_items: "{{ packages\_to\_install | difference(installed\_packages | json\_query('results[\*].name')) }}"

>ansible-playbook var\_users.yml --extra-vars "ansible\_sudo\_pass=usera"

- hosts: localhost

user: ansible

become: True

tasks:

- name: Load passwords from vault

include\_vars: users.yml

- name: Create users

user: name="{{item.key}}" password="{{item.value}}" shell="/bin/bash" update\_password=on\_create

with\_dict: "{{ user\_passwords }}"

register: user\_results

- name: Force password renewal for newly created users

command: chage -d 0 {{item.item.key}} # item.item is the key/value pair from the dict in the previous task

when: "{{item.changed == true}}"

with\_items: "{{ user\_results.results }}

* ansible-playbook var\_groups.yml --extra-vars "ansible\_sudo\_pass=usera"

---

- hosts: localhost

user: ansible

become: True

tasks:

- name: Load groups from file

include\_vars: "groups.yml"

- name: Ensure groups exists

group:

name: "{{ item }}"

state: present

with\_items: "{{ groupes }}"

* + 1. Séparer les tâches d’installation des paquets et création des groupes sur deux fichiers séparés dans un dossier **tasks** et réécrire le playbook avec la balise **include** pour inclure les tâches exportées.

>ansible-playbook grouped\_tasks.yml --extra-vars "ansible\_sudo\_pass=usera"

cat grouped\_tasks.yml

---

- hosts: localhost

user: ansible

become: True

- include: var\_groups.yml

- include: var\_packages.yml

* 1. Etiqueter chaque tâche du playbook en : packages, groupes et users. Ensuite exécuter seulement les 2 premiers tags.

>ansible-playbook playbook.yml –tags "packages, groupes" --extra-vars "ansible\_sudo\_pass=usera"

1. **Gestion des rôles :**

Les rôles sont un niveau d’abstraction supplémentaire dans Ansible. Définir des rôles permet de les combiner par la suite et de « normaliser » la présentation d’un playbook. Nous allons transformer notre playbook en rôle puis voir rapidement Galaxy, le hub d’Ansible où les utilisateurs publient leurs rôles.

* 1. Ecrire un rôle

Faisons un peu de ménage. On va ajouter un répertoire roles qui va stoker les rôles. On crée ensuite un sous-répertoire gestion qui va contenir notre rôle gestion et différents sous répertoires :

mkdir roles

cd roles

mkdir gestion

cd gestion

mkdir files handlers meta templates tasks vars

On va ensuite dans tasks où nous allons mettre la partie des tasks dans un fichier **main.yaml**.

**Première simplification** : dans tasks, il ne doit y avoir que les tâches et plus aucun en-tête. De plus, avec les rôles, on a plus besoin de donner les noms des répertoires pour les fichiers et les templates car ils sont implicites. Les variables seront également mises dans un fichier sous le dossier vars.

Il faut ensuite copier le fichier index.html dans le bon répertoire.

On se replace ensuite au-dessus du répertoire roles et on crée un fichier play.yml Ce fichier va utiliser le role gestion que l’on vient de créer comme suit :

- hosts: suivants become: yes roles:

- role: gestion

On joue ensuite play.yml à l’aide de la commande

ansible-playbook play.yml