# Ansible

***Réalisé par :***

Kawtar Oukil

*Année 2021-2022*

# Table des matières

# Chapitre 1 :

## Introduction

Dans un monde où la technologie évolue constamment à un rythme rapide et croît incroyablement rapidement en même temps, les administrateurs système et les ingénieurs Devops doivent réfléchir à différentes approches sur la façon d'automatiser les tâches de routine et d'orchestrer de grands pools de serveurs. Parmi les approches qu'on peut citer, on retrouve : Ansible.

# 1. Qu'est-ce qu'Ansible ?

Ansible est un outil d'automatisation et d'orchestration open source pour le provisionnement de logiciels, la gestion de la configuration et le déploiement de logiciels. Ansible peut facilement exécuter et configurer des systèmes de type Unix ainsi que des systèmes Windows pour fournir une infrastructure en tant que code. Il contient son propre langage de programmation déclaratif pour la configuration et la gestion du système.

Ansible est populaire pour sa simplicité d'installation, sa facilité d'utilisation en ce qui concerne la connectivité aux clients, son manque d'agent pour les clients Ansible et la multitude de compétences. Il fonctionne en se connectant via SSH aux clients, il n'a donc pas besoin d'un agent spécial côté client, et en poussant les modules vers les clients, les modules sont ensuite exécutés localement côté client et la sortie est repoussée au serveur Ansible.

Puisqu'il utilise SSH, il peut très facilement se connecter aux clients à l'aide de clés SSH, simplifiant ainsi l'ensemble du processus. Les détails du client, comme les noms d'hôte ou les adresses IP et les ports SSH, sont stockés dans des fichiers appelés fichiers d'inventaire. Une fois que vous avez créé un fichier d'inventaire et que vous l'avez rempli, ansible peut l'utiliser.

# 2. Pourquoi utiliser Ansible ?

L'un des avantages les plus importants d'Ansible est qu'il est gratuit pour tout le monde. Il ne nécessite aucune compétence particulière d'administrateur système pour installer et utiliser Ansible, et la documentation officielle est très complète.

Sa modularité concernant les plugins, les modules, les inventaires et les **playbook**s fait d'Ansible le compagnon idéal pour orchestrer de grands environnements.

Ansible est très léger et cohérent, et aucune contrainte concernant le système d'exploitation ou le matériel sous-jacent n'est présente.

Il est également très sécurisé en raison de ses capacités sans agent et de l'utilisation des fonctionnalités de sécurité OpenSSH.

# 3. Histoire d'Ansible

En février 2012, le projet Ansible a commencé. Il a d'abord été développé par Michael DeHaan, le créateur de Cobbler and Func, Fedora Unified Network Controller.

Initialement appelée AnsibleWorks Inc, la société finançant l'outil ansible a été acquise en 2015 par RedHat et plus tard, avec RedHat, elle est passée sous l'égide d'IBM.

À l'heure actuelle, Ansible est inclus dans des distributions telles que Fedora Linux, RHEL, Centos et Oracle Linux.

Un autre avantage qui encourage l'adoption d'Ansible est sa courbe d'apprentissage fluide déterminée par la documentation complète et la structure et la configuration faciles à apprendre.

# 4. Termes importants utilisés dans Ansible

## Serveur ansible :

La machine sur laquelle Ansible est installé et à partir de laquelle toutes les tâches et tous les playbooks seront exécutés

## Module :

Fondamentalement, un module est une commande ou un ensemble de commandes Ansible similaires destinées à être exécutées côté client.

## Task :

Une tâche est une section composée d'une seule procédure à effectuer

## Role :

Un moyen d'organiser les tâches et les fichiers associés à appeler ultérieurement dans un playbook.

## Fact :

Informations extraites du système client à partir des variables globales avec l'opération de collecte des faits

## Inventory :

Fichier contenant des données sur les serveurs client ansible. Défini dans des exemples ultérieurs en tant que fichier hosts

## Play:

Exécution d'un playbook

## Handler:

Tâche appelée uniquement si un notificateur est présent

## Notifier :

Section attribuée à une tâche qui appelle un gestionnaire si la sortie est modifiée

## Tag :

Nom attribué à une tâche qui peut être utilisé ultérieurement pour émettre uniquement cette tâche ou ce groupe de tâches spécifique.

# Chapitre 2 :

## Implémentation

# 1. Installation d'Ansible sous Linux

Une fois que vous avez comparé et pesé vos options et décidé d'opter pour Ansible, l'étape suivante consiste à l'installer sur votre système. Nous passerons en revue les étapes d'installation dans différentes distributions Linux, les plus populaires, dans le prochain petit tutoriel.

## Installation de Ansible sur les systèmes Centos/RedHat

Étape 1) Installation du référentiel EPEL

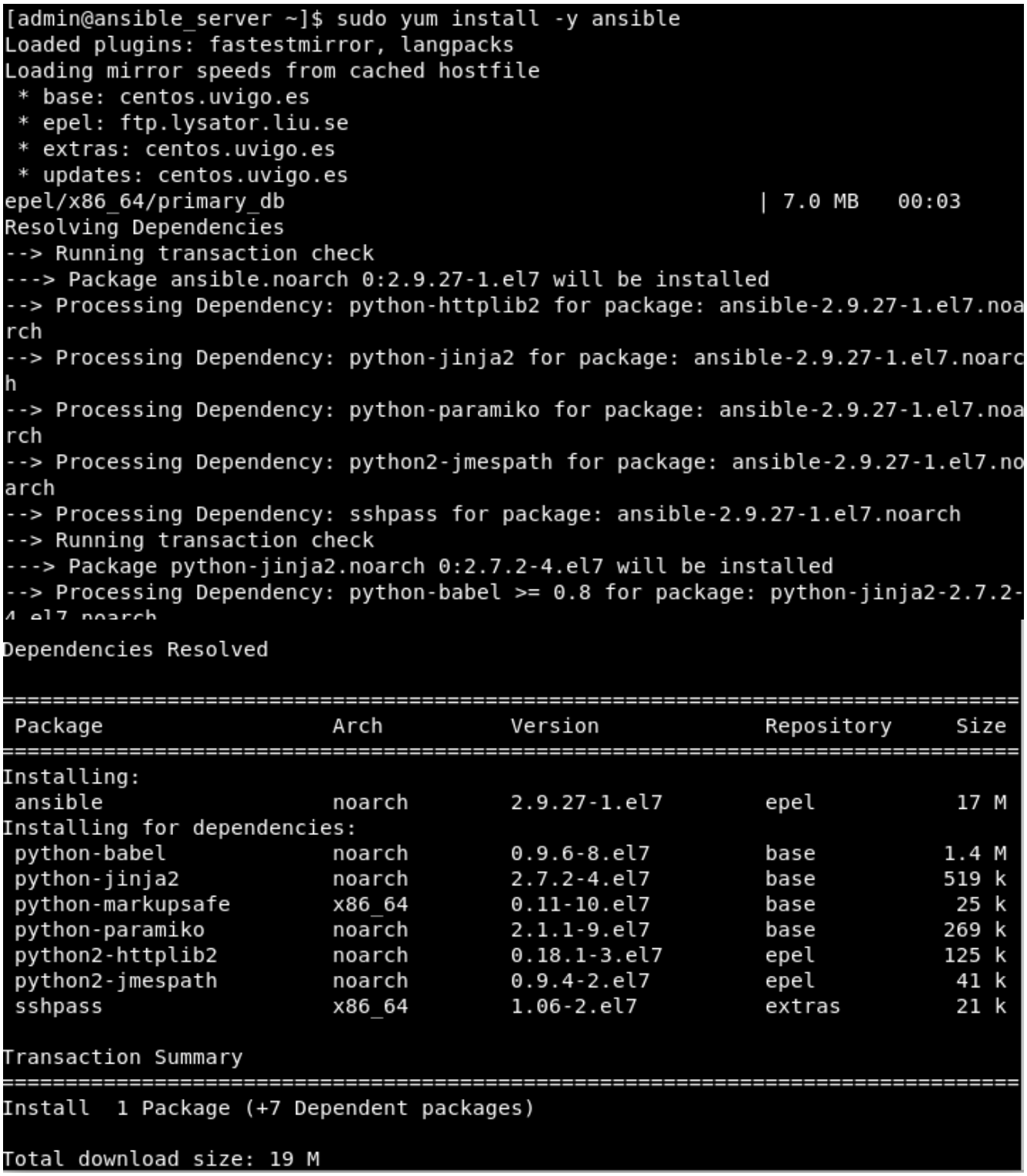
```
[admin@ansible_server ~]$ sudo yum install epel-release
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.uvigo.es
 * extras: centos.uvigo.es
 * updates: centos.uvigo.es
Resolving Dependencies
--> Running transaction check
---> Package epel-release.noarch 0:7-11 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package               Arch             Version         Repository        Size
================================================================================
Installing:
 epel-release          noarch           7-11            extras            15 k

Transaction Summary
================================================================================
Install  1 Package
```

Étape 2) Installation du paquet ansible


[admin@ansible_server ~]$ sudo yum install -y ansible
Loaded plugins: fastestmirror, langpacks
Loading mirror speeds from cached hostfile
 * base: centos.uvigo.es
 * epel: ftp.lysator.liu.se
 * extras: centos.uvigo.es
 * updates: centos.uvigo.es
epel/x86_64/primary_db | 7.0 MB 00:03
Resolving Dependencies
--> Running transaction check
---> Package ansible.noarch 0:2.9.27-1.el7 will be installed
--> Processing Dependency: python-httplib2 for package: ansible-2.9.27-1.el7.noarch
--> Processing Dependency: python-jinja2 for package: ansible-2.9.27-1.el7.noarch
--> Processing Dependency: python-paramiko for package: ansible-2.9.27-1.el7.noarch
--> Processing Dependency: python2-jmespath for package: ansible-2.9.27-1.el7.noarch
--> Processing Dependency: sshpass for package: ansible-2.9.27-1.el7.noarch
--> Running transaction check
---> Package python-jinja2.noarch 0:2.7.2-4.el7 will be installed
--> Processing Dependency: python-babel >= 0.8 for package: python-jinja2-2.7.2-4.el7.noarch
Dependencies Resolved
Package Arch Version Repository Size
Installing:
ansible noarch 2.9.27-1.el7 epel 17 M
Installing for dependencies:
python-babel noarch 0.9.6-8.el7 base 1.4 M
python-jinja2 noarch 2.7.2-4.el7 base 519 k
python-markupsafe x86_64 0.11-10.el7 base 25 k
python-paramiko noarch 2.1.1-9.el7 base 269 k
python2-httplib2 noarch 0.18.1-3.el7 epel 125 k
python2-jmespath noarch 0.9.4-2.el7 epel 41 k
sshpass x86_64 1.06-2.el7 extras 21 k
Transaction Summary
Install 1 Package (+7 Dependent packages)
Total download size: 19 M

# 2. Configuration d'Ansible :

Pour pouvoir utiliser les commandes, un simple fichier d'hôtes à deux serveurs doit être configuré, contenant **server**1 et **server**2.

```
[admin@ansible_server ~]$ cd /etc/ansible
[admin@ansible_server ansible]$ ls
ansible.cfg  hosts  roles
```

On ajoute au fichier ***/etc/ansible/hosts*** les lignes suivantes **:**

```
[web-servers]
192.168.229.142 ansible_ssh_username=root ansible_ssh_pass=ansible
192.168.229.139 ansible_ssh_username=root ansible_ssh_pass=ansible
```

# 3. Les Commandes Ansible Ad Hoc

L'une des façons les plus simples d'utiliser Ansible consiste à utiliser des commandes ad hoc. Ceux-ci peuvent être utilisés lorsque on souhaite émettre des commandes sur un serveur ou un groupe de serveurs. Les commandes ad hoc ne sont pas stockées pour des utilisations futures mais représentent un moyen rapide d'interagir avec les serveurs souhaités.

Voyons maintenant quelques exemples d'implémentation des commandes Ansible :

- *Exemple 1 :* Accessibilité des hôtes :

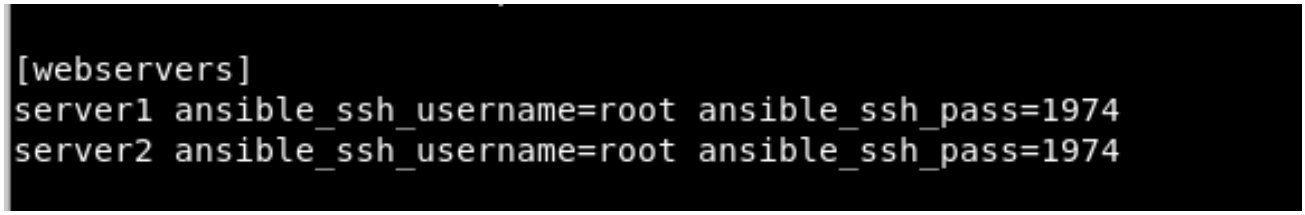
On peut s'assurer que les hôtes sont accessibles depuis le serveur ansible en émettant une commande ping sur tous les hôtes


Explication :

1. Statut de la commande, dans ce cas SUCCESS
2. Hôte sur lequel la commande a été exécutée
3. La commande émise via le paramètre -m, dans ce cas, ping

On peut même remplacer les adresses IP des serveurs par leur hostname de cette façon :

On modifie le fichier ***/etc/ansible/hosts*** :



Et en laçant la commande précédente, on obtient :

```
[root@ansible_server ansible]# ansible all -m ping
server1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
server2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

On peut émettre la même commande uniquement sur un hôte spécifique si nécessaire, on spécifiant son nom dans la commande :

```
[root@ansible_server ansible]# ansible server1 -m ping
server1 | SUCCESS => {                          1
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Explication :

1. Nom de l'hôte tel que défini dans le fichier d'inventaire.

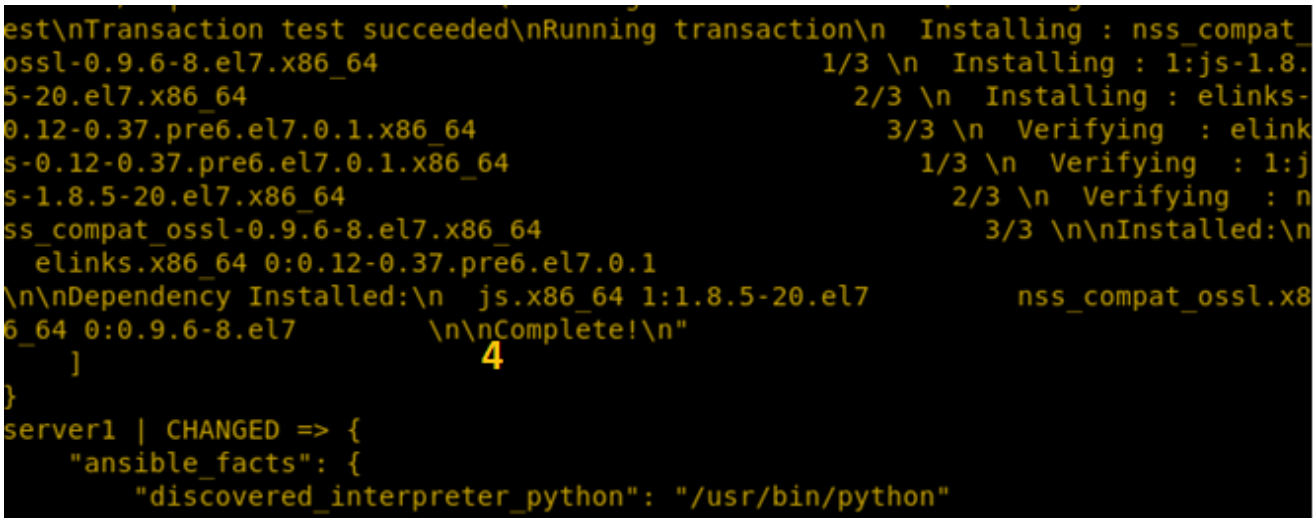
- ***Exemple 2 :*** Affichage de la liste des fichiers :

Dans le cas où on a besoin d'afficher rapidement la liste des fichiers de plusieurs hosts, on peut utiliser la commande « ls » dans ansible. Ainsi, la commande et sa sortie ressemblent à ci-dessous :

```
[root@ansible_server ansible]# ansible all -a "ls /"
server2 | CHANGED | rc=0 >>
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
server1 | CHANGED | rc=0 >>
bin
boot
dev
etc
home
lib
lib64
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
var
```

- ***Exemple 3 :*** Installation d'un package :

Si on veut installer un package , on peut utiliser le module **yum** sur deux hôtes Centos

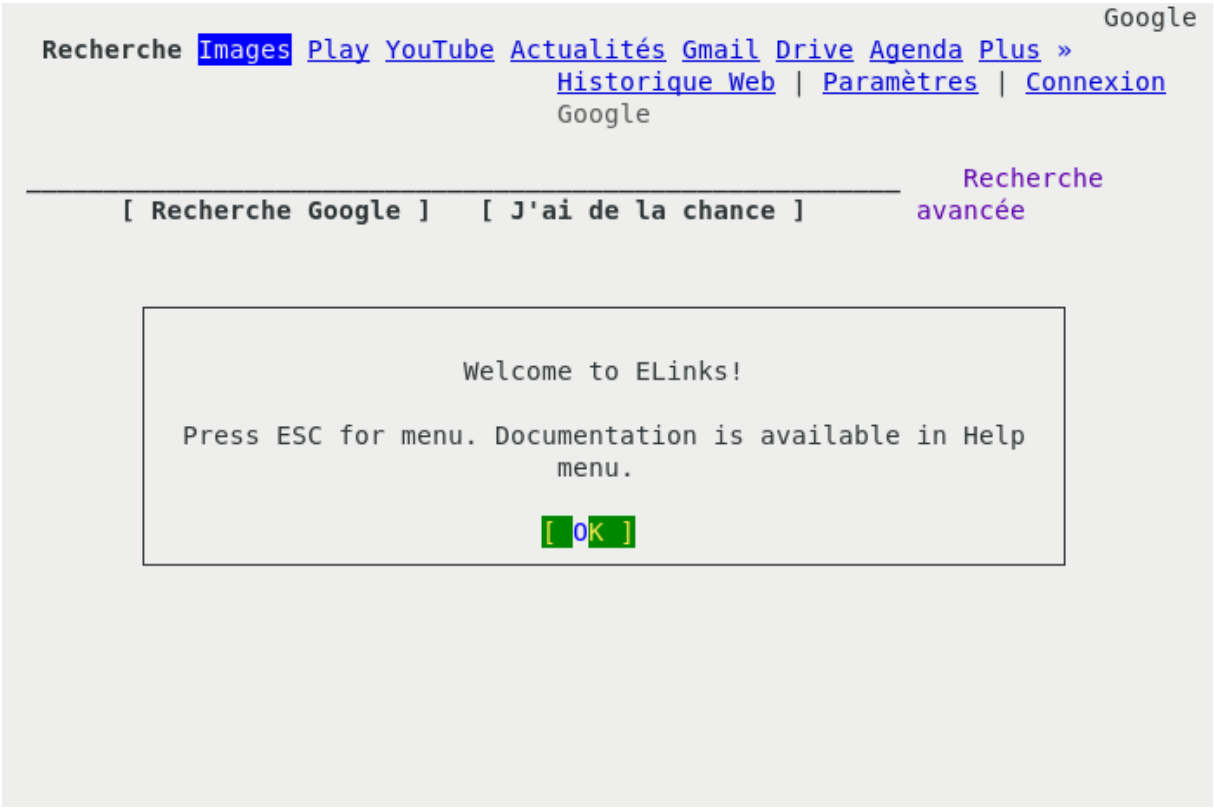



Explication :

1. Le module Yum est utilisé dans cet exemple
2. Il définit les arguments du module, et dans ce cas, vous choisirez le nom du package et son état. Si l'état est absent, par exemple, le paquet sera recherché et s'il est trouvé, supprimé
3. Lorsqu'il est coloré en jaune, vous verrez la sortie de la commande ansible avec l'état modifié, ce qui signifie dans ce cas que le paquet a été trouvé et installé.
4. État de la commande yum install émise via ansible. Dans ce cas, le package « elinks » a été installé.

Et pour tester l'installation de ce package, on peut exécuter la commande ci-dessous :

```
[root@server2 ~]# elinks google.ca
```




- ***Exemple 4*** : Suppression d'un package :

Bien sûr, toutes les options d'installation de yum peuvent être utilisées via ansible, y compris la mise à jour, l'installation, la dernière version ou la suppression.

Dans l'exemple ci-dessous, la même commande a été émise pour supprimer le package « elinks » précédemment installé.


Explication :

1. La sortie de la commande yum indique que le package a été supprimé.

- *Exemple 5 :* Affichage des informations détaillées :

Une autre fonctionnalité utile et essentielle qu'ansible utilise pour interagir avec le serveur du client est de rassembler quelques faits sur le système.

Ainsi, il récupère les informations sur le matériel, les logiciels et les versions du système et stocke chaque valeur dans une variable qui peut être utilisée ultérieurement.

Si vous avez besoin d'informations détaillées sur les systèmes à modifier via ansible, la commande suivante peut être utilisée. Le module de configuration rassemble des faits à partir des variables système.


# 4. Playbooks d'Ansible :

Les Playbooks Ansible sont le moyen d'envoyer des commandes à des systèmes distants via des scripts. Les playbooks Ansible sont utilisés pour configurer des environnements système complexes afin d'augmenter la flexibilité en exécutant un script sur un ou plusieurs systèmes.

Les playbooks Ansible ont tendance à être plus un langage de configuration qu'un langage de programmation.

Les commandes de playbook Ansible utilisent le format YAML, il n'y a donc pas beaucoup de syntaxe nécessaire, mais l'indentation doit être respectée. Comme son nom l'indique, un

playbook est une collection de jeux. Grâce à un playbook, on peut attribuer des rôles spécifiques à certains hôtes et d'autres rôles à d'autres hôtes. Par ce fait, on peut orchestrer plusieurs serveurs dans des scénarios très divers, le tout dans un seul playbook.

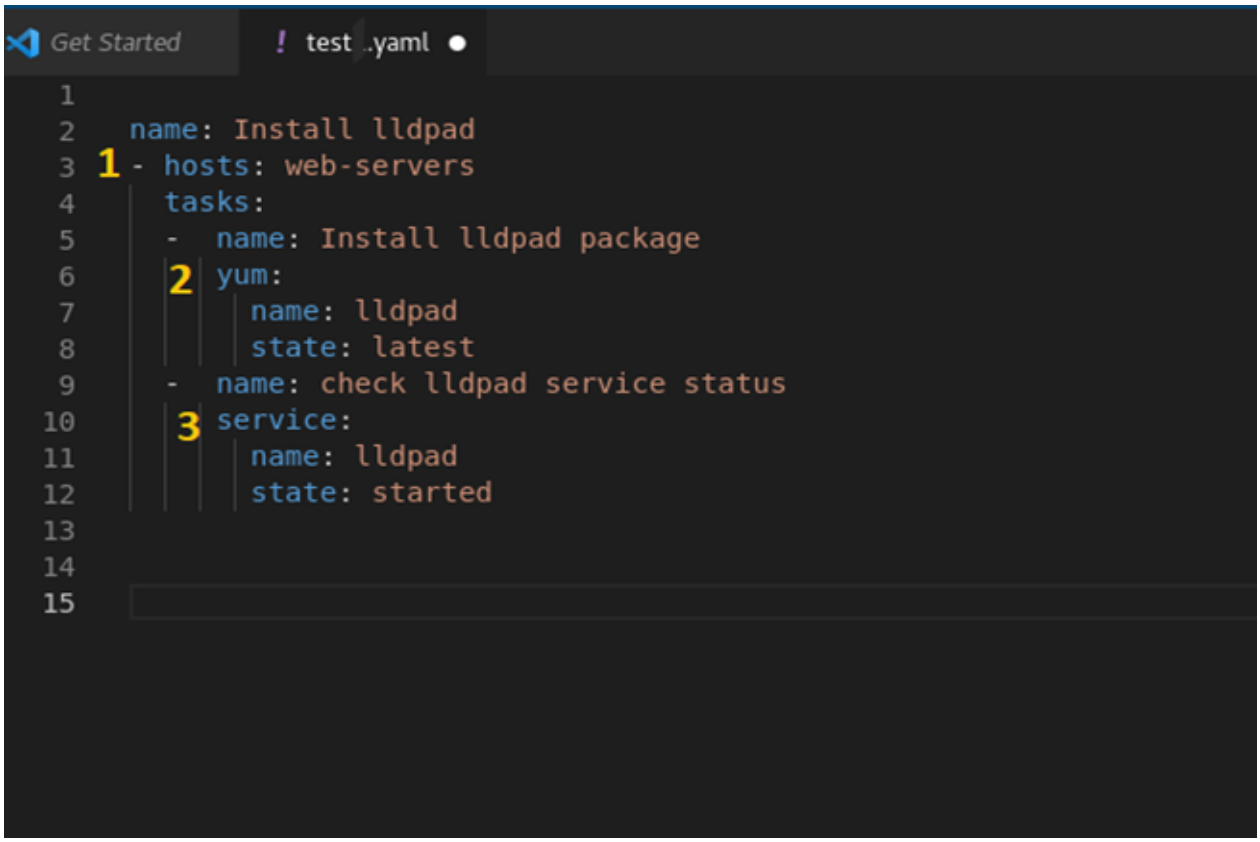
Pour avoir tous les détails précis avant de continuer avec des exemples de playbook Ansible, il faut d'abord définir une tâche. Il s'agit de l'interface des modules ansible pour les roles et les playbooks.

Chaque playbook ansible fonctionne avec un fichier d'inventaire. Le fichier d'inventaire contient une liste de serveurs divisés en groupes pour un meilleur contrôle des détails tels que l'adresse IP et le port SSH pour chaque hôte. Pour notre cas, ces informations des serveurs sont stockées dans le fichier ***/etc/ansible/hosts***

```
[webservers]
server1 ansible_ssh_username=root ansible_ssh_pass=1974
server2 ansible_ssh_username=root ansible_ssh_pass=1974
```

- *Exemple 1* : Installation et démarrage d'un package :

Maintenant, apprenons le playbook Ansible à travers un exemple avec un playbook avec un Play, contenant plusieurs tâches comme ci-dessous :

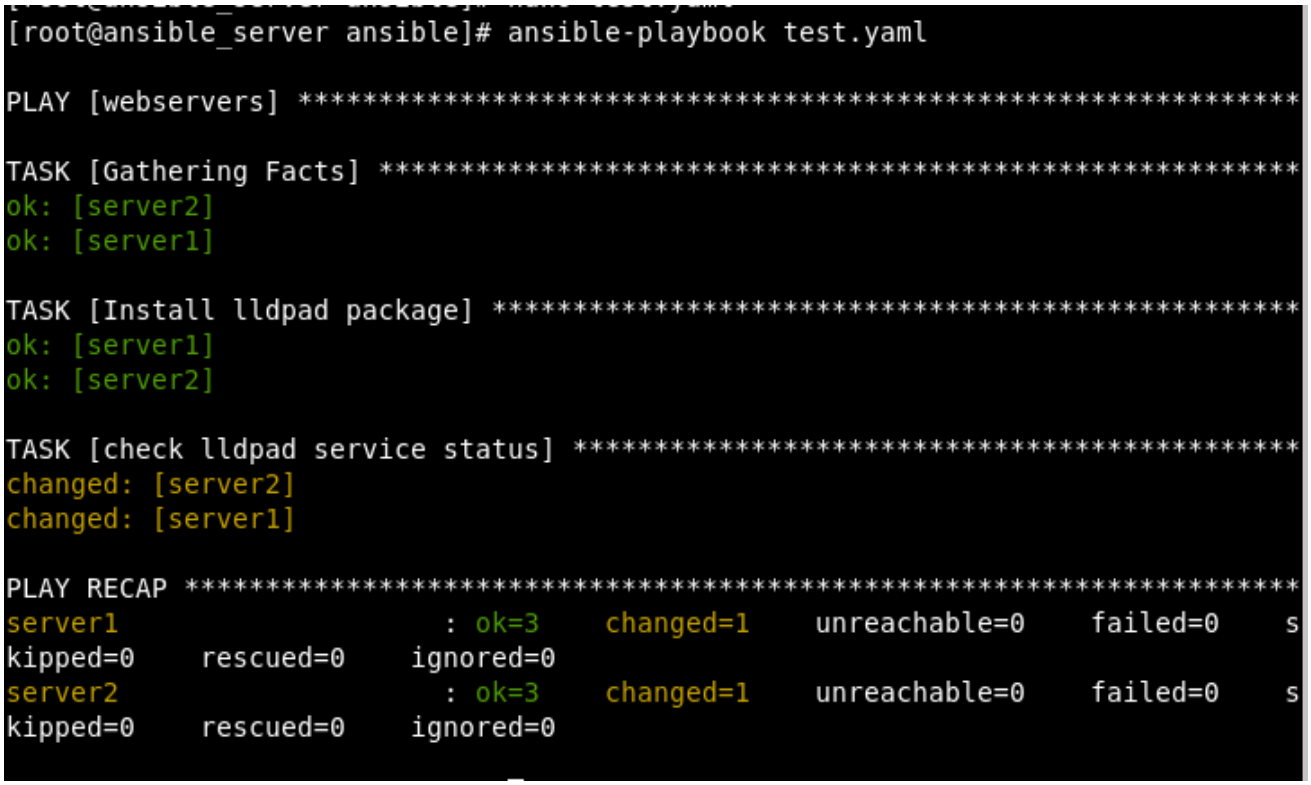



Dans l'exemple de playbook Ansible ci-dessus, l'hôte 1 est ciblé pour l'installation du package « lldpad » à l'aide du module **yum**, puis le service lldpad créé après l'installation est ensuite démarré à l'aide du module de **service** utilisé principalement pour interagir avec l'ensemble système.

Explication :

1. Groupe d'hôtes sur lesquels le playbook sera exécuté
2. Le module Yum est utilisé dans cette tâche pour l'installation de lldpad
3. Le module de service est utilisé pour vérifier si le service est opérationnel après l'installation

Et en exécutant la commande ***ansible-playbook*** on obtient :



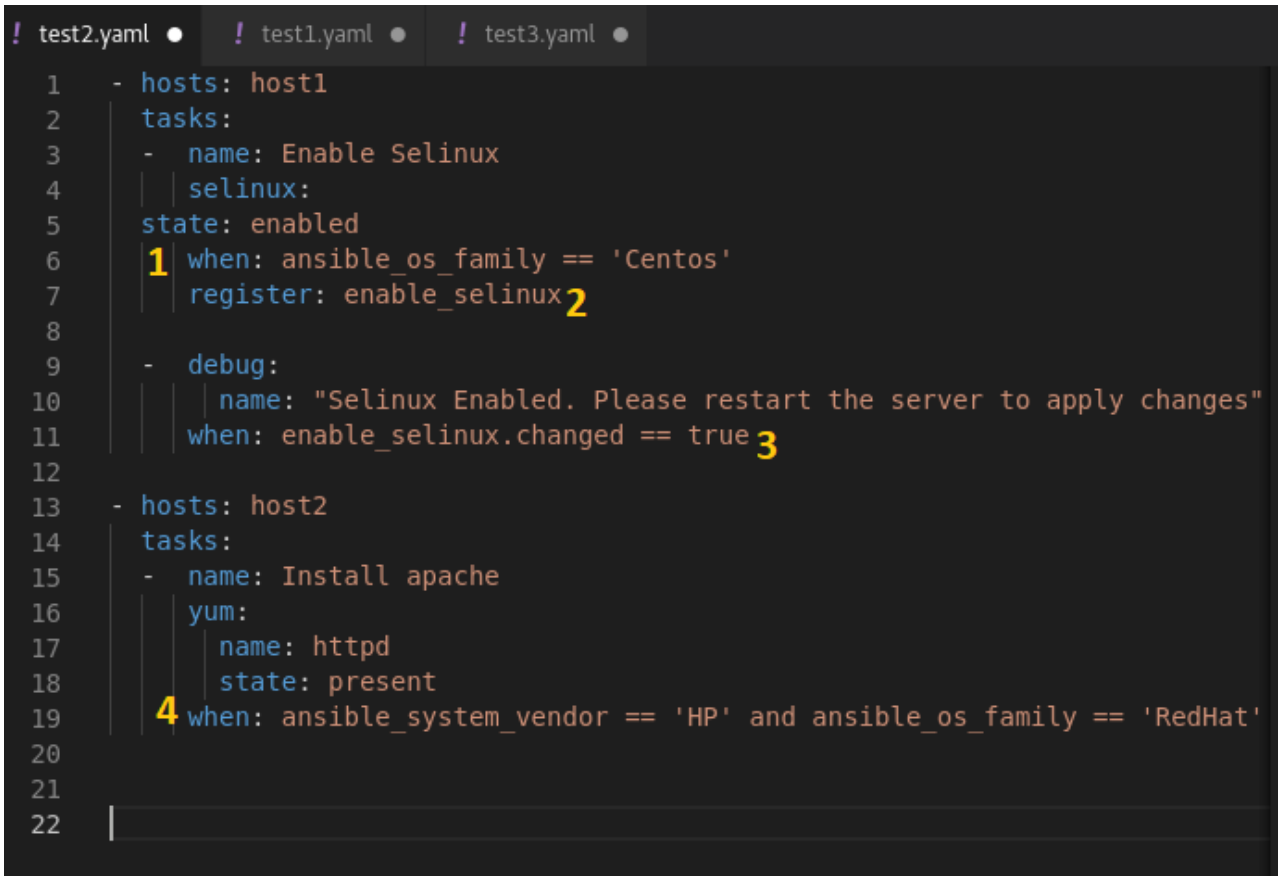

- *Exemple 2* : Installation et activation de deux packages différents :

Un autre exemple utile de playbook Ansible contenant cette fois deux jeux pour deux hôtes est le suivant. Pour le premier hôte, **selinux** sera activé. S'il est activé, un message apparaîtra sur l'écran de l'hôte.

Pour le deuxième hôte, le package **httpd** sera installé uniquement si *ansible_os_family* est RedHat et *ansible_system_vendor* est HP.

*Ansible_os_family* et *ansible_system_vendor* sont des variables rassemblées avec l'option *rassemble_facts* et peuvent être utilisées comme dans cet exemple conditionnel.

Le fichier playbook ***test2.yaml:***




Explication :

1. Exemple de clause ***when***, Dans ce cas, lorsque le type de système d'exploitation est Debian. La variable ansible_os_family est collectée via la fonctionnalité de *rassemblement_facts*.
2. La sortie de la tâche est enregistrée pour une utilisation future, avec son nom *enable_selinux*
3. Un autre exemple de la clause **when**. Dans ce cas, un message s'affichera pour l'utilisateur hôte si le SELinux était bien activé auparavant.
4. Un autre exemple de la clause **when** composée de deux règles

En exécutant ce playbook on obtient le résultat ci-dessous :

```
[root@ansible_server ansible]# ansible-playbook test2.yaml

PLAY [host1] ***********************************************************************

TASK [Gathering Facts] *************************************************************
ok: [server1]

TASK [Enable Selinux] **************************************************************
skipping: [server1]

TASK [debug] ***********************************************************************
skipping: [server1]

PLAY [host2] ***********************************************************************

TASK [Gathering Facts] *************************************************************
ok: [server2]

TASK [Install apache] **************************************************************
skipping: [server2]

PLAY RECAP *************************************************************************
server1                    : ok=1    changed=0    unreachable=0    failed=0    s
kipped=2    rescued=0    ignored=0
```

- *Exemple 3 :* Installation et démarrage d'un package et la création d'un page html :

Le fichier ***/etc/ansible/hosts*** :

```
[webservers]
192.168.120.134 ansible_ssh_username=root ansible_ssh_pass=Mery
192.168.120.138 ansible_ssh_username=root ansible_ssh_pass=Mery
```

Le fichier playbook ***test.yaml :***

test.yaml ● | test1.yaml ● | test2.yaml ● | test3.yaml ●

```yaml
- hosts: web-servers
  remote_user: root
  become: true
  connection: ssh
  tasks:
  - name: Install HTTPD server on target servers
    yum:
      name: httpd
      state: latest



  - name: Start HTTPD server on target servers
    service:
      name: httpd
      state: started

  - name: Create basic index.html file in the apache root dir
    copy:
    - name: Start HTTPD server on target servers
      service:
        name: httpd
        state: started

    - name: Create basic index.html file in the apache root dir
      copy:
        content: "This is server-ansible Test Page"
        dest: /var/www/html/index.html
```

Et en exécutant ce playbook on obtient le résultat ci-dessous :

```
[root@ausible ansible]# ansible-playbook test.yaml

PLAY [HTTPD Book] ***********************************************************************************************************

TASK [Gathering Facts] ******************************************************************************************************
ok: [192.168.120.138]
ok: [192.168.120.134]

TASK [Install HTTPD server on target servers] ********************************************************************************
changed: [192.168.120.138]
changed: [192.168.120.134]

TASK [start HTTPD service on target servers] *********************************************************************************
changed: [192.168.120.134]
changed: [192.168.120.138]

TASK [create basic index.html file in the apache root dir] *******************************************************************
changed: [192.168.120.138]
changed: [192.168.120.134]

PLAY RECAP ******************************************************************************************************************
192.168.120.134            : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
192.168.120.138            : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Pour se confirmer du résultat, on a choisi de tester l'etat du package sur les deux serveurs :

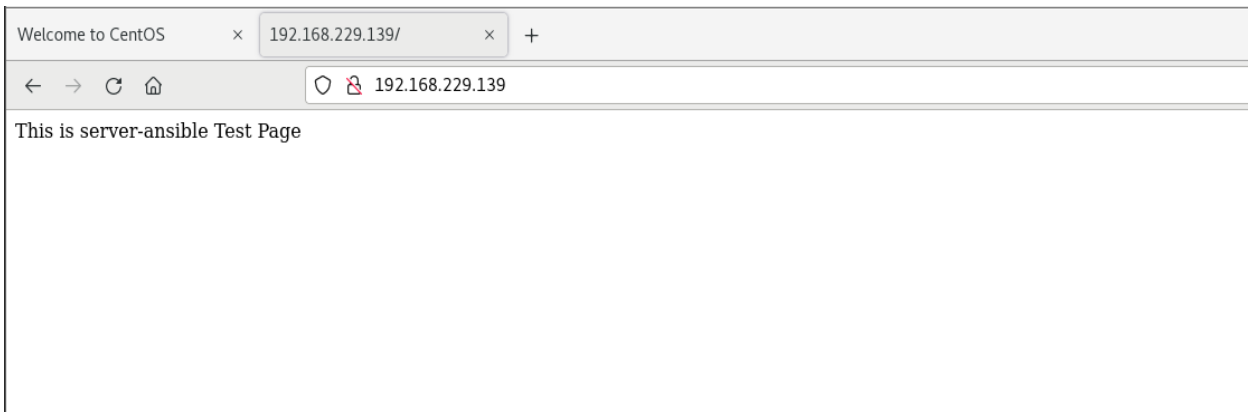
```
[root@server1 mery]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; vendor pre
bled)
   Active: active (running) since Thu 2022-06-16 17:01:50 PDT; 54s ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 3786 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffic:   0 E
    Tasks: 6
   CGroup: /system.slice/httpd.service
           ├─3786 /usr/sbin/httpd -DFOREGROUND
           ├─3791 /usr/sbin/httpd -DFOREGROUND
           ├─3792 /usr/sbin/httpd -DFOREGROUND
           ├─3793 /usr/sbin/httpd -DFOREGROUND
           ├─3794 /usr/sbin/httpd -DFOREGROUND
           └─3795 /usr/sbin/httpd -DFOREGROUND

Jun 16 17:01:50 server1 systemd[1]: Starting The Apache HTTP Server...
Jun 16 17:01:50 server1 httpd[3786]: AH00558: httpd: Could not reliably determ
Jun 16 17:01:50 server1 systemd[1]: Started The Apache HTTP Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

```
[root@server2 mery]# systemctl status httpd
● httpd.service - The Apache HTTP Server
   Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; ve
bled)
   Active: active (running) since Thu 2022-06-16 17:01:50 PDT; 39s ago
     Docs: man:httpd(8)
           man:apachectl(8)
 Main PID: 3749 (httpd)
   Status: "Total requests: 0; Current requests/sec: 0; Current traffi
    Tasks: 6
   CGroup: /system.slice/httpd.service
           ├─3749 /usr/sbin/httpd -DFOREGROUND
           ├─3754 /usr/sbin/httpd -DFOREGROUND
           ├─3755 /usr/sbin/httpd -DFOREGROUND
           ├─3756 /usr/sbin/httpd -DFOREGROUND
           ├─3757 /usr/sbin/httpd -DFOREGROUND
           └─3758 /usr/sbin/httpd -DFOREGROUND

Jun 16 17:01:50 server2 systemd[1]: Starting The Apache HTTP Server...
Jun 16 17:01:50 server2 httpd[3749]: AH00558: httpd: Could not reliabl
Jun 16 17:01:50 server2 systemd[1]: Started The Apache HTTP Server.
Hint: Some lines were ellipsized, use -l to show in full.
```

Concernant la page html, on peut l'afficher en tapant l'adresse IP de l'hôte et on obtient le message affiché dans la page :



# 5.Roles d'Ansible

Lorsqu'il s'agit de playbooks volumineux, il est plus facile de diviser les tâches en roles. Cela aide également à réutiliser les roles à l'avenir. Les roles sont une collection de tâches, qui peuvent être déplacées d'un playbook à un autre, peuvent être exécutées indépendamment mais uniquement via un fichier playbook.

Les roles sont stockés dans des répertoires séparés et ont une structure de répertoire particulière.

Pour créer l'arborescence de répertoires pour un role, vous devez utiliser la commande suivante avec le dernier paramètre, le nom du role :

```
[root@ansible_server ansible]# cd roles
[root@ansible_server roles]# ansible-galaxy init role1
- Role role1 was created successfully
```

```
[root@ansible_server roles]# tree
.
└── role1
    ├── defaults
    │   └── main.yml
    ├── files
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── README.md
    ├── tasks
    │   └── main.yml
    ├── templates
    ├── tests
    │   ├── inventory
    │   └── test.yml
    └── vars
        └── main.yml

9 directories, 8 files
```

tasks/main.yml :la liste principale des tâches que le rôle exécute.

handlers/main.yml : handlers, qui peuvent être utilisés à l'intérieur ou à l'extérieur de ce rôle.

defaults/main.yml : variables par défaut pour le rôle

vars/main.yml : autres variables pour le rôle

files/main.yml : fichiers que le rôle déploie.

templates/main.yml : modèles déployés par le rôle.

meta/main.yml : métadonnées du rôle, y compris les dépendances de role.

Le répertoire tests contient un exemple de fichier de playbook yaml et un exemple de fichier d'inventaire et est principalement utilisé à des fins de test avant de créer le rôle réel.

Pour créer un role, il faut accéder au répertoire ***/etc/ansible/roles/role1/tasks*** pour remplir le fichier **main.yml**

```
[root@ansible_server ansible]# cd roles
[root@ansible_server roles]# ll
total 0
drwxr-xr-x 10 root root 154 Jun 22 08:14 role1
[root@ansible_server roles]# cd role1
[root@ansible_server role1]# ll
total 4
drwxr-xr-x 2 root root   22 Jun 22 08:14 defaults
drwxr-xr-x 2 root root    6 Jun 22 08:14 files
drwxr-xr-x 2 root root   22 Jun 22 08:14 handlers
drwxr-xr-x 2 root root   22 Jun 22 08:14 meta
-rw-r--r-- 1 root root 1328 Jun 22 08:14 README.md
drwxr-xr-x 2 root root   43 Jun 22 09:10 tasks
drwxr-xr-x 2 root root    6 Jun 22 08:14 templates
drwxr-xr-x 2 root root   39 Jun 22 08:14 tests
drwxr-xr-x 2 root root   22 Jun 22 08:14 vars
[root@ansible_server role1]# cd tasks
[root@ansible_server tasks]# ll
total 8
-rw-r--r-- 1 root root 26 Jun 22 08:14 main.yml
-rw-r--r-- 1 root root 80 Jun 22 09:10 main.yml.save
[root@ansible_server tasks]# nano main.yml
```

Role1 :

```
  GNU nano 2.3.1        File: roles/role1/tasks/main.yml


- name: Install Apache  package
  yum:
    name: python3
    state: latest
```

```
[root@ansible_server ansible]# ll
total 36
-rw-r--r-- 1 root root 19985 Jun 21 09:05 ansible.cfg
-rw-r--r-- 1 root root  1272 Jun 22 06:04 hosts
drwxr-xr-x 3 root root    19 Jun 22 08:14 roles
-rw-r--r-- 1 root root   471 Jun 22 06:19 test2.yaml
-rw-r--r-- 1 root root   466 Jun 22 06:44 test3.yaml
-rw-r--r-- 1 root root   210 Jun 21 16:02 test.yaml
[root@ansible_server ansible]# nano test-role1.yaml
```

Et pour exécuter le role, on remplit le fichier ***test-role1.yaml***

```
  GNU nano 2.3.1            File: test-role1.yaml

- hosts: host1
  roles:
  - { role: role1 }
```

Et en exécutant la commande ***ansible-playbook*** on obtient :

```
[root@ansible_server ansible]# ansible-playbook test-role1.yaml

PLAY [host1] *******************************************************************

TASK [Gathering Facts] *********************************************************
ok: [server1]

TASK [role1 : Install Apache  package] *****************************************
changed: [server1]

PLAY RECAP *********************************************************************
server1                    : ok=2    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0
```

# 6.Étude des cas Ansible

Dans cette section, nous analyserons une étude de cas d'un playbook ansible essentiel qui a trois rôles. Le but de ceci est de donner un exemple pratique de ce dont nous avons parlé précédemment. Certains des exemples utilisés auparavant dans ce didacticiel de playbook Ansible seront adaptés et utilisés dans ce playbook.

La structure des répertoires du playbook sera défini dans le fichier ***test-roles.yaml***.

```
  GNU nano 2.3.1            File: test-roles.yaml

- hosts: host2
  roles:
  - { role: role1 }
  - { role: role2 }
  - { role: role3 }
```

Le playbook a trois roles, un appelé **role1** qui installe le package « python3 ». Un autre s'appelle **role2**, et il installe le package « ntp » avec le module yum, et le troisième installe le package « kernel ». Chaque rôle a été créé à l'aide de la commande ansible-galaxy.

```
[root@ansible_server ansible]# cd roles
[root@ansible_server roles]# ansible-galaxy init role1
- Role role1 was created successfully
```

```
[root@ansible_server roles]# ansible-galaxy init role2
- Role role2 was created successfully
```

```
[root@ansible_server roles]# ansible-galaxy init role3
- Role role3 was created successfully
```

Donc on obtient :

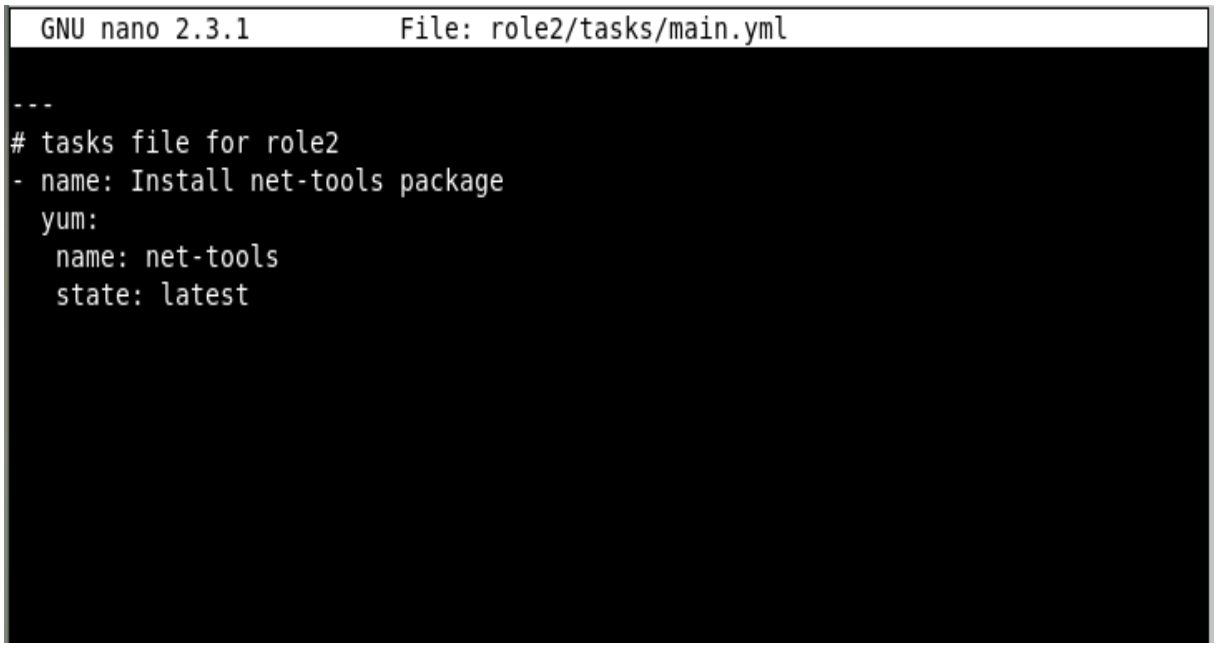
```
[root@ansible_server ansible]# cd roles
[root@ansible_server roles]# ll
total 0
drwxr-xr-x 10 root root 154 Jun 22 08:14 role1
drwxr-xr-x 10 root root 154 Jun 22 10:20 role2
drwxr-xr-x 10 root root 154 Jun 22 10:20 role3
```

Role 1, tâche main.yml :

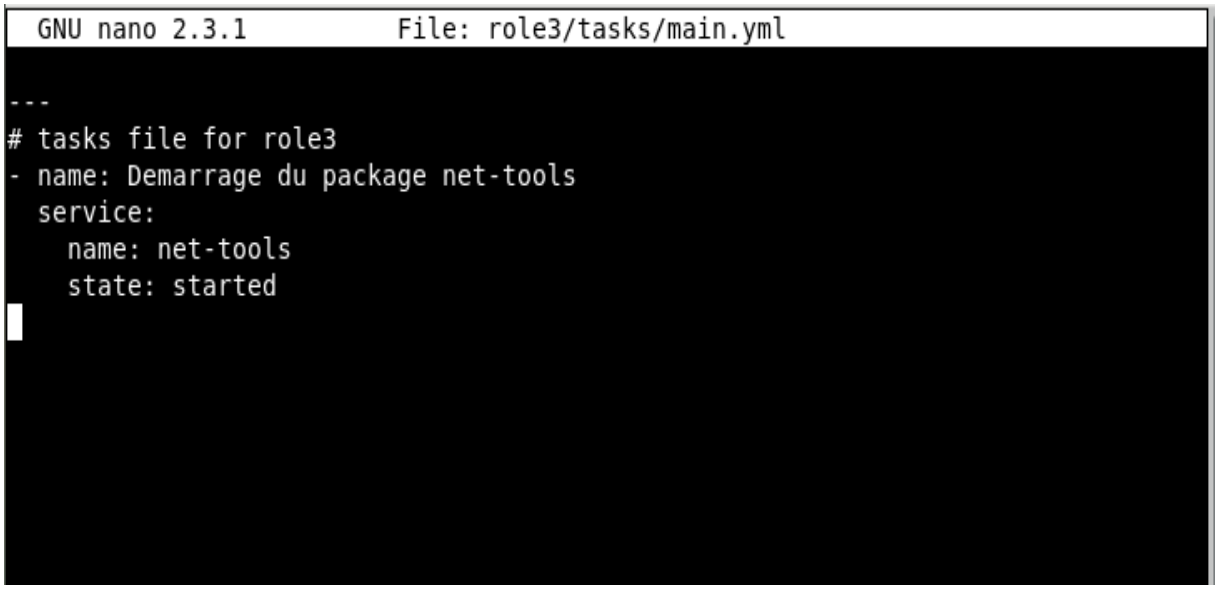
```
 GNU nano 2.3.1          File: role1/tasks/main.yml


- name: Install Apache  package
  yum:
    name: python3
    state: latest
```

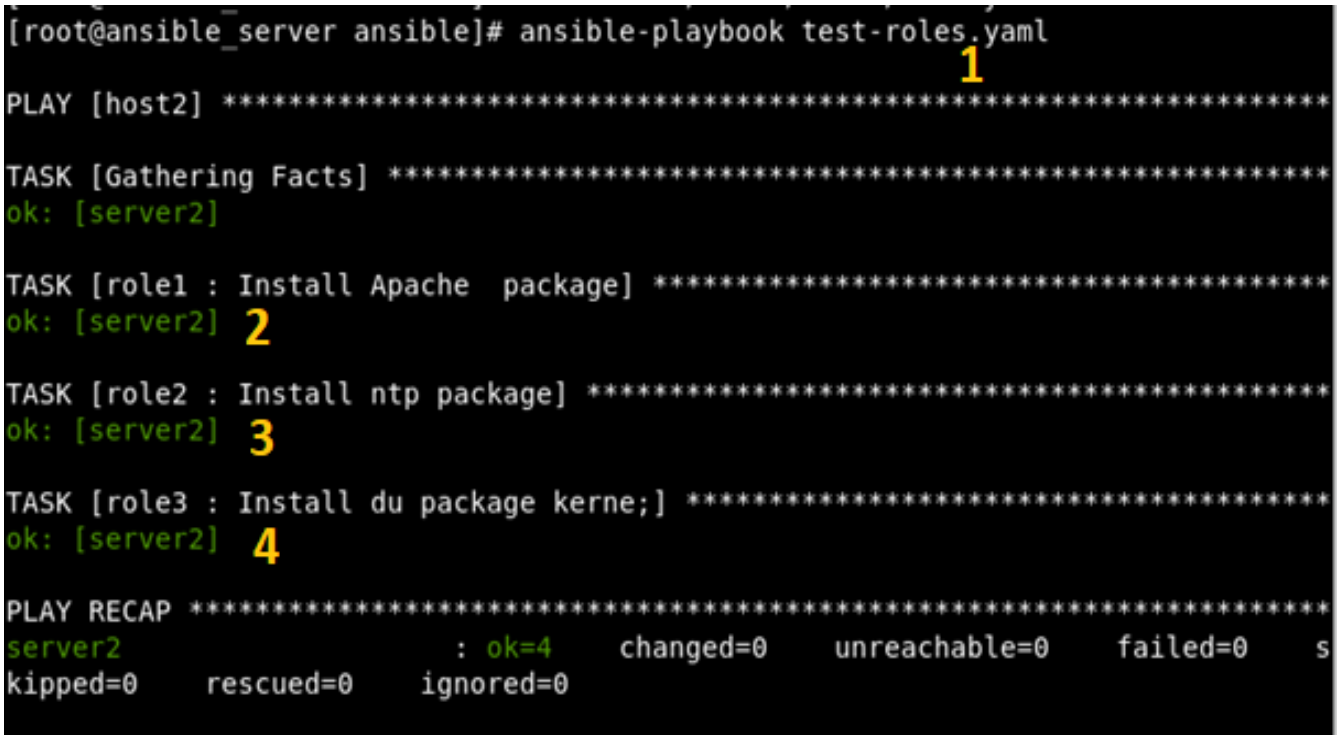
Role 2, tâche main.yml :




Role 3, tâche main.yml :




Et en exécutant le playbook ***test-roles.yaml*** sur le deuxième hôte , on obtient le résultat suivant :




Explication :

1. Commande ansible-playbook qui exécute ***test-roles.yaml***
2. Ansible a trouvé que le package « python3 » est déjà installé, il renvoie donc ok.
3. Ansible a trouvé que le package « ntp » est déjà installé, il renvoie donc ok.
4. Ansible a trouvé que le package « kernel » est déjà installé, il renvoie donc ok.

# Conclusion

Bien qu'il existe de nombreuses alternatives à Ansible (Chef, Puppet) qui font la même chose avec quelques différences, Ansible a réussi à les dépasser grâce à sa simplicité, sa sécurité améliorée et, surtout, sa courbe d'apprentissage fluide. En raison de ces qualités et de l'adoption rapide d'Ansible, nous avons créé un didacticiel rempli d'exemples afin que vous puissiez avoir une première expérience encore plus fluide avec Ansible.

Dans ce rapport sur les bases d'Ansible, nous avons décrit ansible et parlé un peu de son histoire. Nous avons évoqué les points forts d'Ansible et les avantages qu'Ansible peut apporter à l'automatisation et à l'orchestration d'infrastructures de différentes tailles. Nous avons défini les termes essentiels utilisés par ansible et défini la structure des playbooks Ansible. Des exemples détaillés accompagnaient toutes les informations d'explications détaillées.