NATIONAL SCHOOL OF APPLIED SCIENCES
KHOURIBGA

SECTOR: (I.R.I.C)

SCHOOL YEAR: 2021-2022

KINGDOM OF MOROCCO

ADMISTRATION RESEAUX

project report

THEME : ANSIBLE

PRESENTED BY :

Rharif Anass

UNDER THE GUIDANCE OF

Dr Yassine MALEH

ANSIBLE PROJECT REPORT

DEFINITION

Ansible is a configuration management tool that allows to automate tasks with automation scripts.

In this project we will use ansible to automate the following steps:

1. Install an Apache web server on the first server.
2. Install PHP also on the first server.
3. Install a MariaDB database on the second server.

(source : https://openclassrooms.com/).

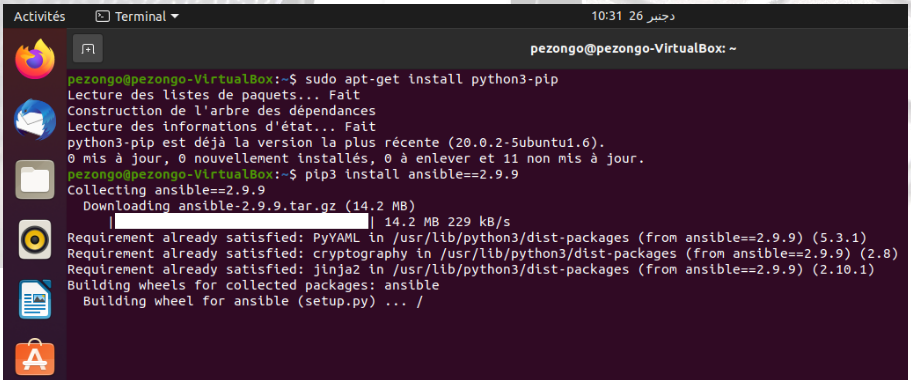
I. ANSIBLE INSTALLATION

We can install ansible by installation packages, by pip or do a binary installation.

Here we will use pip installation with the following commands:

sudo apt-get install python3-pip

pip3 install ansible==2.9.9



```
pezongo@pezongo-VirtualBox:~$ sudo apt-get install python3-pip
Lecture des listes de paquets... Fait
Construction de l'arbre des dépendances
Lecture des informations d'état... Fait
python3-pip est déjà la version la plus récente (20.0.2-5ubuntu1.6).
0 mis à jour, 0 nouvellement installés, 0 à enlever et 11 non mis à jour.
pezongo@pezongo-VirtualBox:~$ pip3 install ansible==2.9.9
Collecting ansible==2.9.9
  Downloading ansible-2.9.9.tar.gz (14.2 MB)
     |████████████████████████████████| 14.2 MB 229 kB/s
Requirement already satisfied: PyYAML in /usr/lib/python3/dist-packages (from ansible==2.9.9) (5.3.1)
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (from ansible==2.9.9) (2.8)
Requirement already satisfied: jinja2 in /usr/lib/python3/dist-packages (from ansible==2.9.9) (2.10.1)
Building wheels for collected packages: ansible
  Building wheel for ansible (setup.py) ... /
```

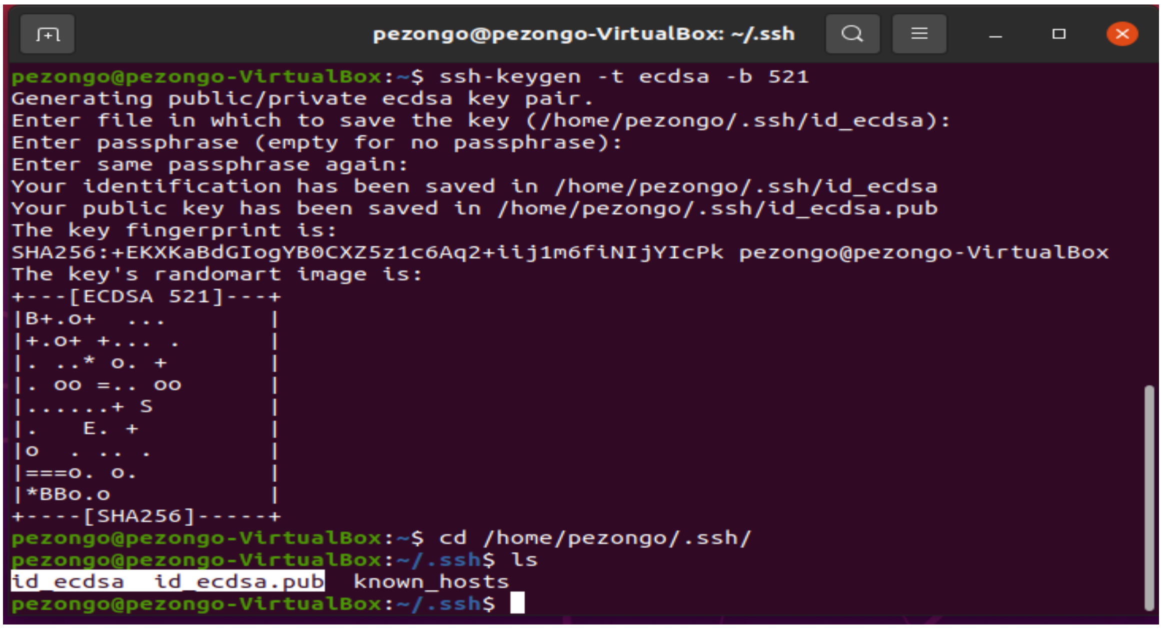
II. SSH KEY GENERATION

To use ansible we will generate a SSH key that will allow us to connect to our servers in a secure way. We will define a public key and a private key with the following commands:

ssh-keygen -t ecdsa -b 521

We use -b 521 to define the size of the key.

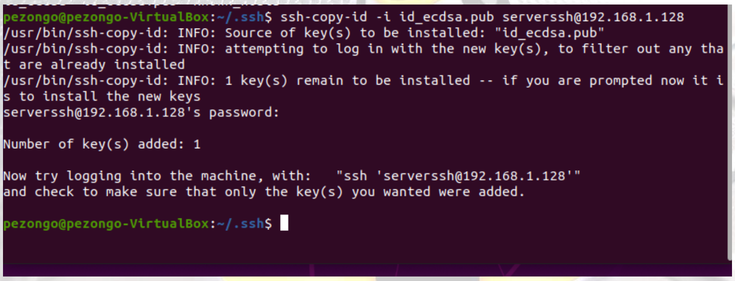
In the following picture we can see that the public and private key have been created



```
pezongo@pezongo-VirtualBox:~$ ssh-keygen -t ecdsa -b 521
Generating public/private ecdsa key pair.
Enter file in which to save the key (/home/pezongo/.ssh/id_ecdsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/pezongo/.ssh/id_ecdsa
Your public key has been saved in /home/pezongo/.ssh/id_ecdsa.pub
The key fingerprint is:
SHA256:+EKXKaBdGIogYB0CXZ5z1c6Aq2+iij1m6fiNIjYIcPk pezongo@pezongo-VirtualBox
The key's randomart image is:
+---[ECDSA 521]---+
|B+.o+   ...      |
|+.o+ +... .      |
|. ..* o. +       |
|. oo =.. oo      |
|......+ S        |
|.    E. +        |
|o   . .. .       |
|===o. o.         |
|*BBo.o           |
+----[SHA256]-----+
pezongo@pezongo-VirtualBox:~$ cd /home/pezongo/.ssh/
pezongo@pezongo-VirtualBox:~/.ssh$ ls
id_ecdsa  id_ecdsa.pub  known_hosts
pezongo@pezongo-VirtualBox:~/.ssh$ 
```

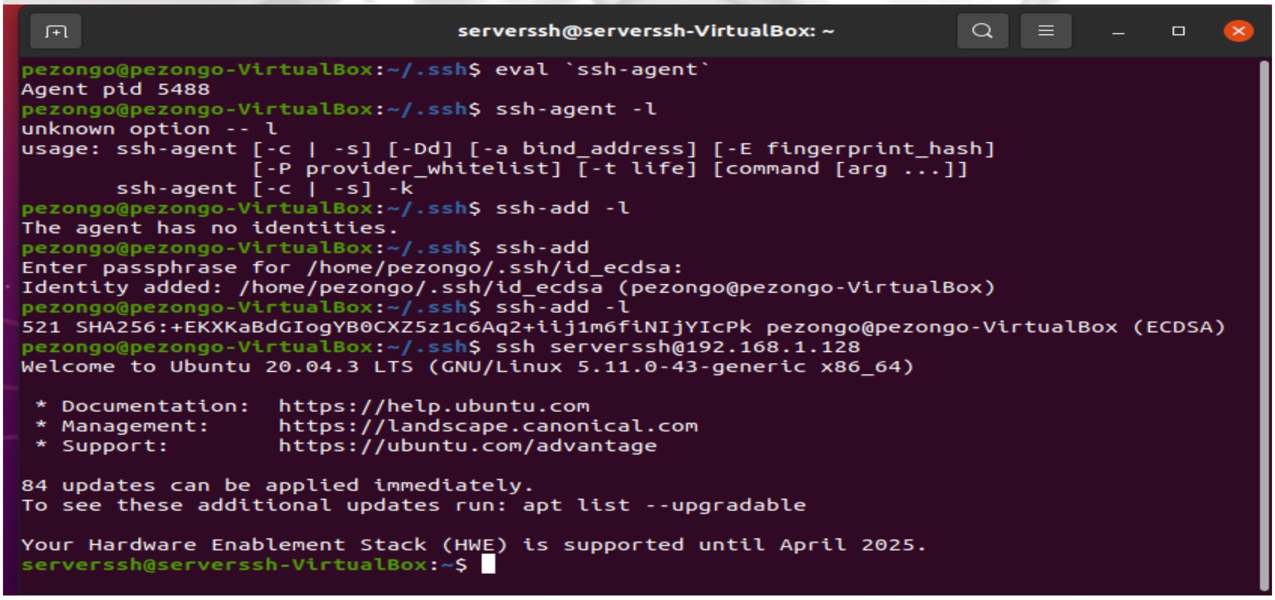
Now we will send the key to the server with the command:

ssh-copy-id -i id_ecdsa.pub serverssh@ip



```
pezongo@pezongo-VirtualBox:~/.ssh$ ssh-copy-id -i id_ecdsa.pub serverssh@192.168.1.128
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "id_ecdsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any tha
t are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it i
s to install the new keys
serverssh@192.168.1.128's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'serverssh@192.168.1.128'"
and check to make sure that only the key(s) you wanted were added.

pezongo@pezongo-VirtualBox:~/.ssh$ 
```

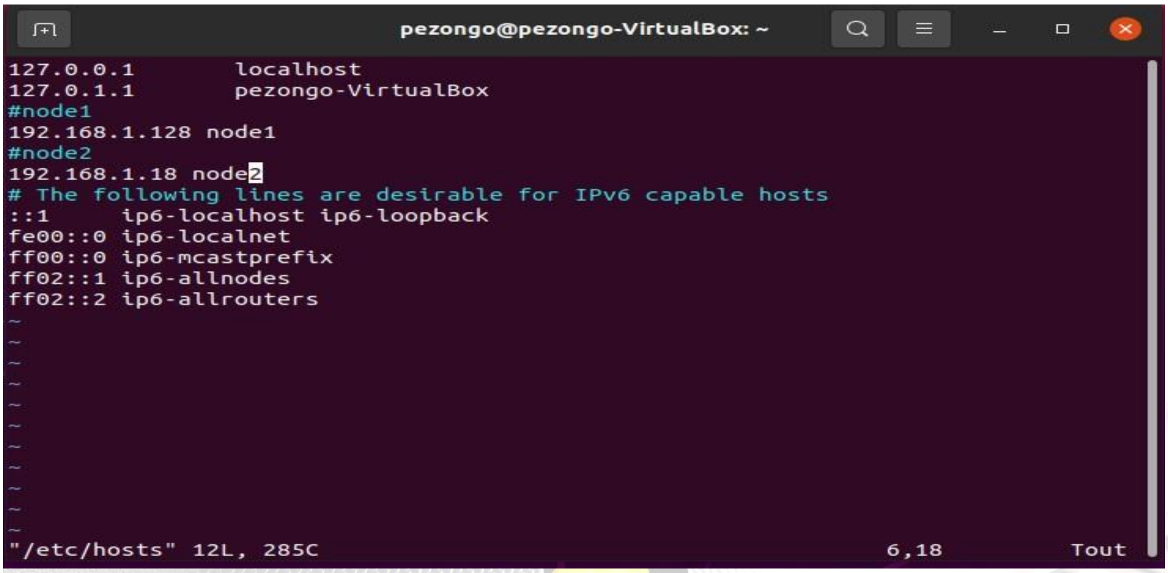
Pour se connecter plus facilement aux serveurs avec ssh, on va créer un agent et y intégrer la clef ssh :



```
pezongo@pezongo-VirtualBox:~/.ssh$ eval `ssh-agent`
Agent pid 5488
pezongo@pezongo-VirtualBox:~/.ssh$ ssh-agent -l
unknown option -- l
usage: ssh-agent [-c | -s] [-Dd] [-a bind_address] [-E fingerprint_hash]
                 [-P provider_whitelist] [-t life] [command [arg ...]]
       ssh-agent [-c | -s] -k
pezongo@pezongo-VirtualBox:~/.ssh$ ssh-add -l
The agent has no identities.
pezongo@pezongo-VirtualBox:~/.ssh$ ssh-add
Enter passphrase for /home/pezongo/.ssh/id_ecdsa:
Identity added: /home/pezongo/.ssh/id_ecdsa (pezongo@pezongo-VirtualBox)
pezongo@pezongo-VirtualBox:~/.ssh$ ssh-add -l
521 SHA256:+EKXKaBdGIogYB0CXZ5z1c6Aq2+iij1m6fiNIjYIcPk pezongo@pezongo-VirtualBox (ECDSA)
pezongo@pezongo-VirtualBox:~/.ssh$ ssh serverssh@192.168.1.128
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.11.0-43-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

84 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Your Hardware Enablement Stack (HWE) is supported until April 2025.
serverssh@serverssh-VirtualBox:~$ 
```

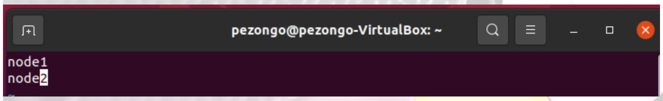
III. COMMUNICATION WITH SERVERS

For an easier communication with the servers we will edit the /etc/hosts file to add their ip address

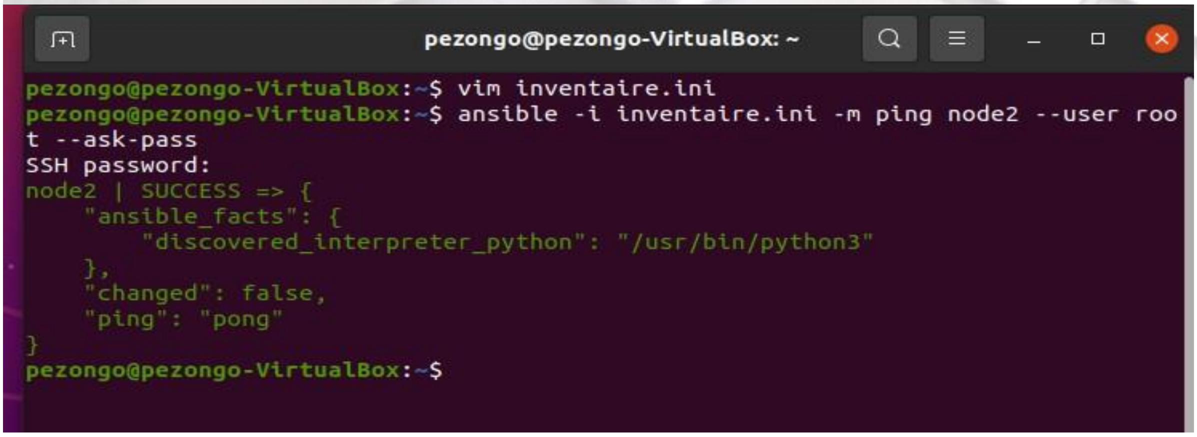


```
127.0.0.1       localhost
127.0.1.1       pezongo-VirtualBox
#node1
192.168.1.128 node1
#node2
192.168.1.18 node2
# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
~
~
~
~
~
~
~
~
~
~
~
"/etc/hosts" 12L, 285C                                        6,18          Tout
```

We create an inventaire.ini file to put the name of the nodes. Then we will launch a ping with ansible to check the connectivity:



```
node1
node2
```



```
pezongo@pezongo-VirtualBox:~$ vim inventaire.ini
pezongo@pezongo-VirtualBox:~$ ansible -i inventaire.ini -m ping node2 --user roo
t --ask-pass
SSH password:
node2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
pezongo@pezongo-VirtualBox:~$
```

Connectivity works.

IV. DEPLOYMENT OPERATIONS

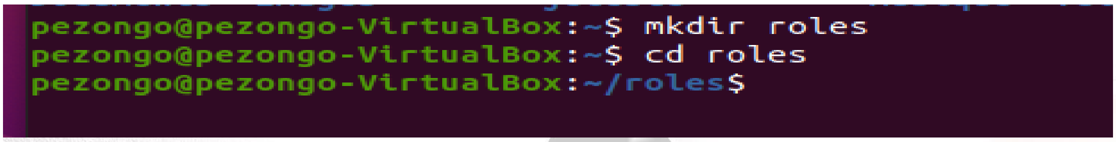
Now we will organize the deployment using ansible roles. A role is therefore a set of files organized in a tree structure. The purpose of the roles is to be able to execute a set of tasks that have the same objective such as installing apache2 or installing Mariadb. A task is a statement described in YAML in a configuration file.

YAML (Yet Another Markup Language). YAML allows to write data structures that can be specified as lists.

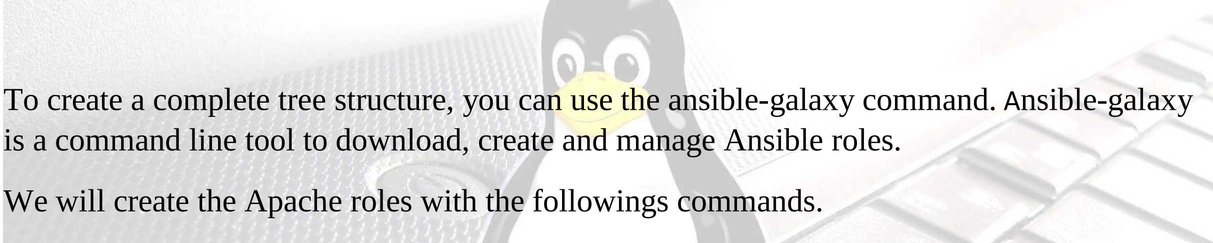
We will create 2 roles:

1. A role to install Apache : apache.
2. A role to install MariaDB: mariadb.

We will create the roles directory where ansible will look for all the roles.

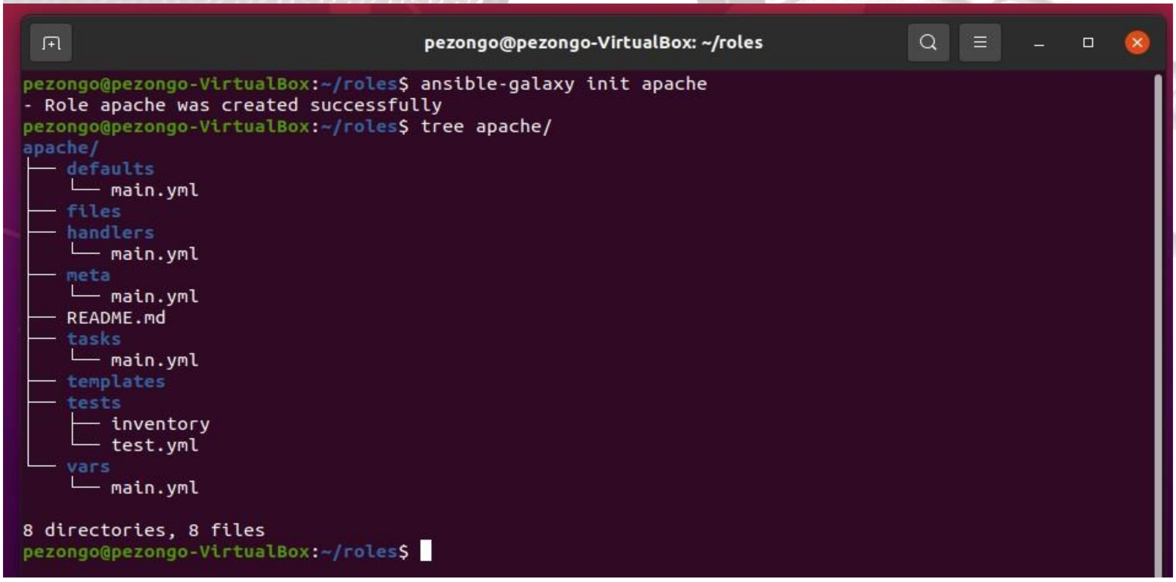


```
pezongo@pezongo-VirtualBox:~$ mkdir roles
pezongo@pezongo-VirtualBox:~$ cd roles
pezongo@pezongo-VirtualBox:~/roles$
```



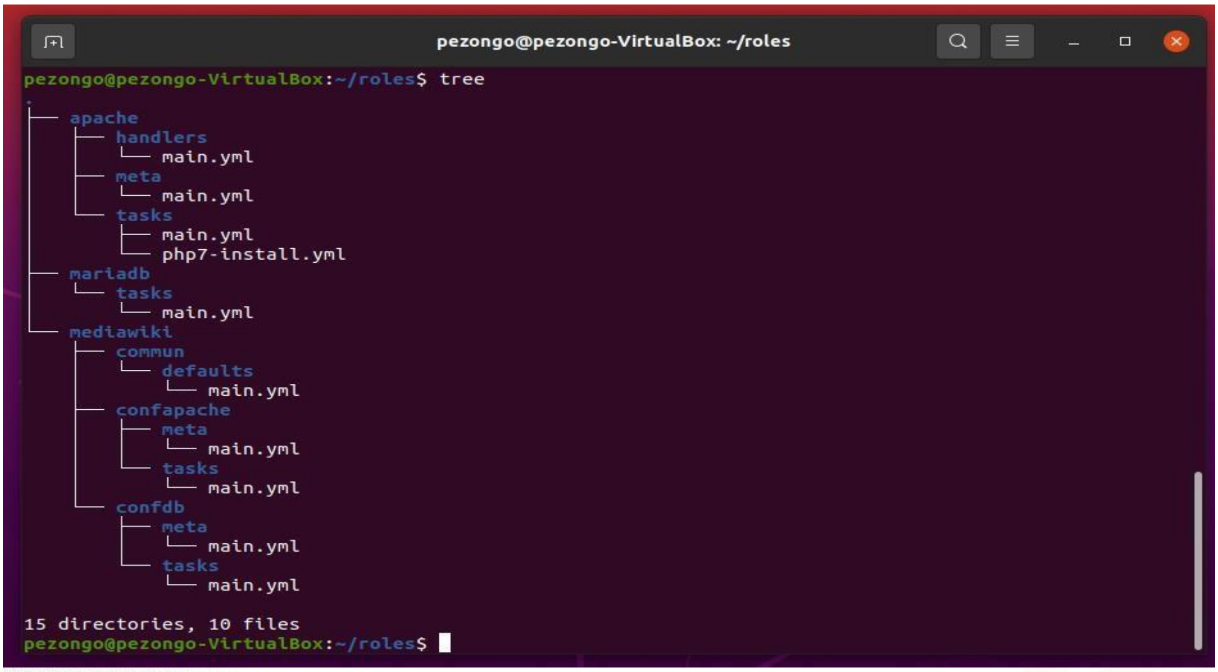
To create a complete tree structure, you can use the ansible-galaxy command. Ansible-galaxy is a command line tool to download, create and manage Ansible roles.

We will create the Apache roles with the followings commands.



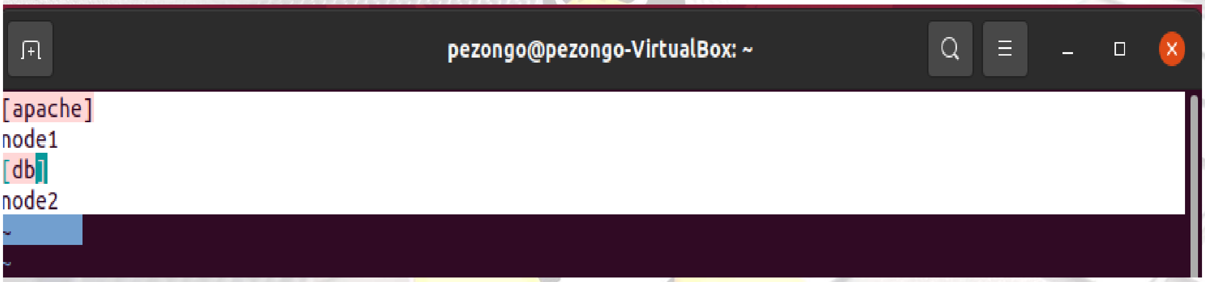
```
pezongo@pezongo-VirtualBox:~/roles$ ansible-galaxy init apache
- Role apache was created successfully
pezongo@pezongo-VirtualBox:~/roles$ tree apache/
apache/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml

8 directories, 8 files
pezongo@pezongo-VirtualBox:~/roles$ 
```

The tree contains several directories but we just need the handler’s directory (which contains the tasks to execute after a notification), tasks (which contains the tasks to execute to install Apache) and meta. So, we will delete the other directories. Then we will create the other directories that we need, which will give the following tree structure:



```
pezongo@pezongo-VirtualBox:~/roles$ tree
.
├── apache
│   ├── handlers
│   │   └── main.yml
│   ├── meta
│   │   └── main.yml
│   └── tasks
│       ├── main.yml
│       └── php7-install.yml
├── mariadb
│   └── tasks
│       └── main.yml
└── mediawiki
    ├── commun
    │   └── defaults
    │       └── main.yml
    ├── confapache
    │   ├── meta
    │   │   └── main.yml
    │   └── tasks
    │       └── main.yml
    └── confdb
        ├── meta
        │   └── main.yml
        └── tasks
            └── main.yml

15 directories, 10 files
pezongo@pezongo-VirtualBox:~/roles$
```

Finally, we will add the servers in specific groups by modifying the inventaire.ini file as follows:



```
[apache]
node1
[db]
node2
~
~
```

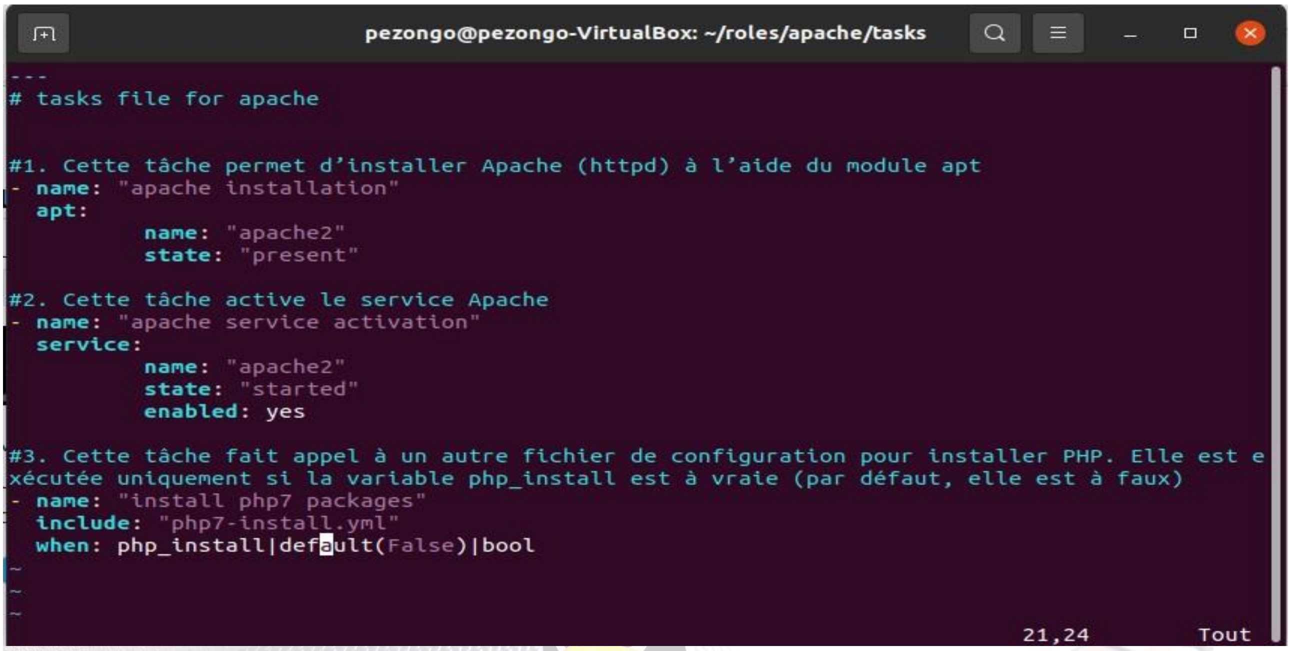
We have just finished the creation of our files for the deployment of our tasks. Now we will start writing the automation scripts.

V. Automation scripts

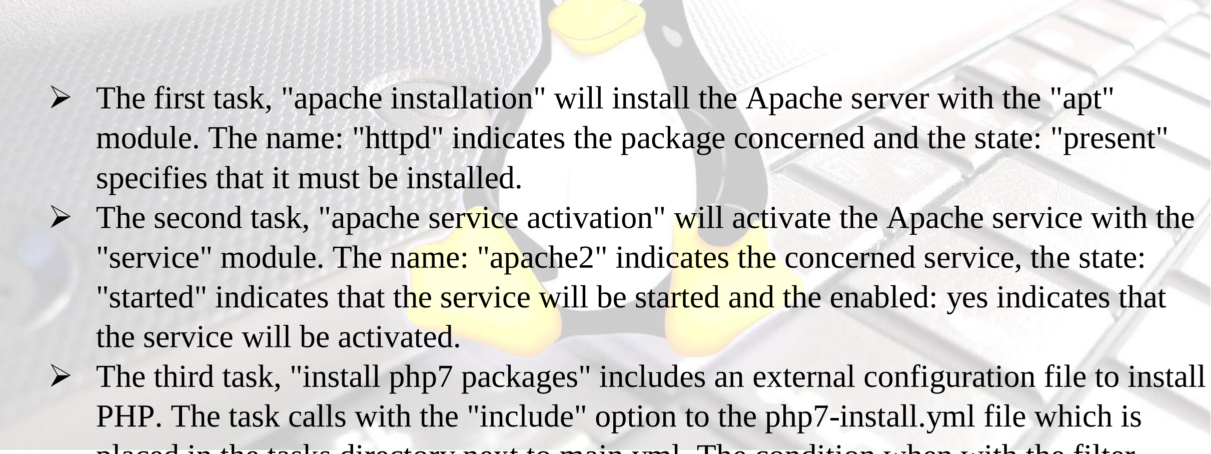
We will build the automation scripts by completing the main.yml files present in each role. This will allow us to execute tasks and link several actions. Following our previous tree structure, we will create 2 scripts in YAML file for the 2 roles.

1. **The apache role:**
    - a tasks/main.yml file contains actions to install Apache, a call to a configuration file to install PHP, and a notification to restart Apache;
    - a tasks/php7-install.yml file contains the actions to install PHP;
    - a handler/main.yml file contains the actions to restart the Apache service.
2. **The mariadb role:**
    - A tasks/main.yml file contains the actions to install MariaDB.

YAML files always start with 3 dashes (---). Then you have the different successive tasks that start with 1 dash (-) and the task name. Each task uses a module with its arguments or options. The arguments or options are offset on the line by 2 spaces.

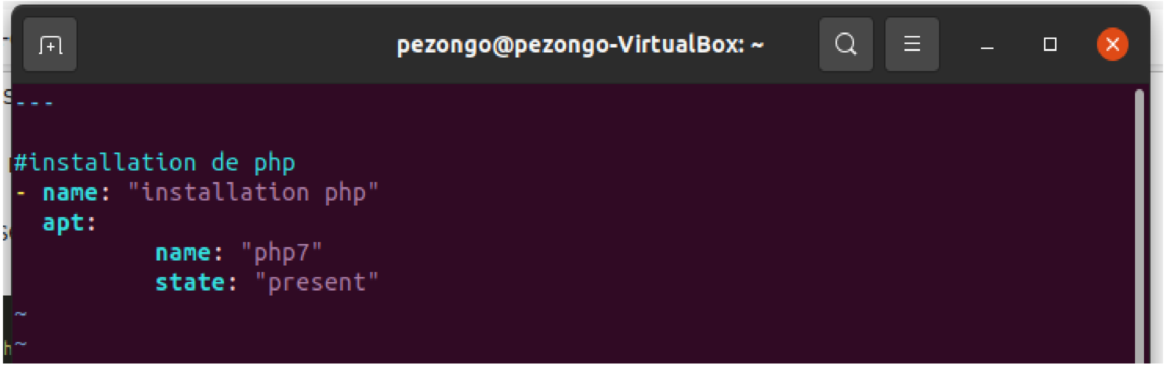


```
---
# tasks file for apache


#1. Cette tâche permet d'installer Apache (httpd) à l'aide du module apt
- name: "apache installation"
  apt:
          name: "apache2"
          state: "present"

#2. Cette tâche active le service Apache
- name: "apache service activation"
  service:
          name: "apache2"
          state: "started"
          enabled: yes

#3. Cette tâche fait appel à un autre fichier de configuration pour installer PHP. Elle est e
xécutée uniquement si la variable php_install est à vraie (par défaut, elle est à faux)
- name: "install php7 packages"
  include: "php7-install.yml"
  when: php_install|default(False)|bool
~
~
~
                                                                         21,24         Tout
```



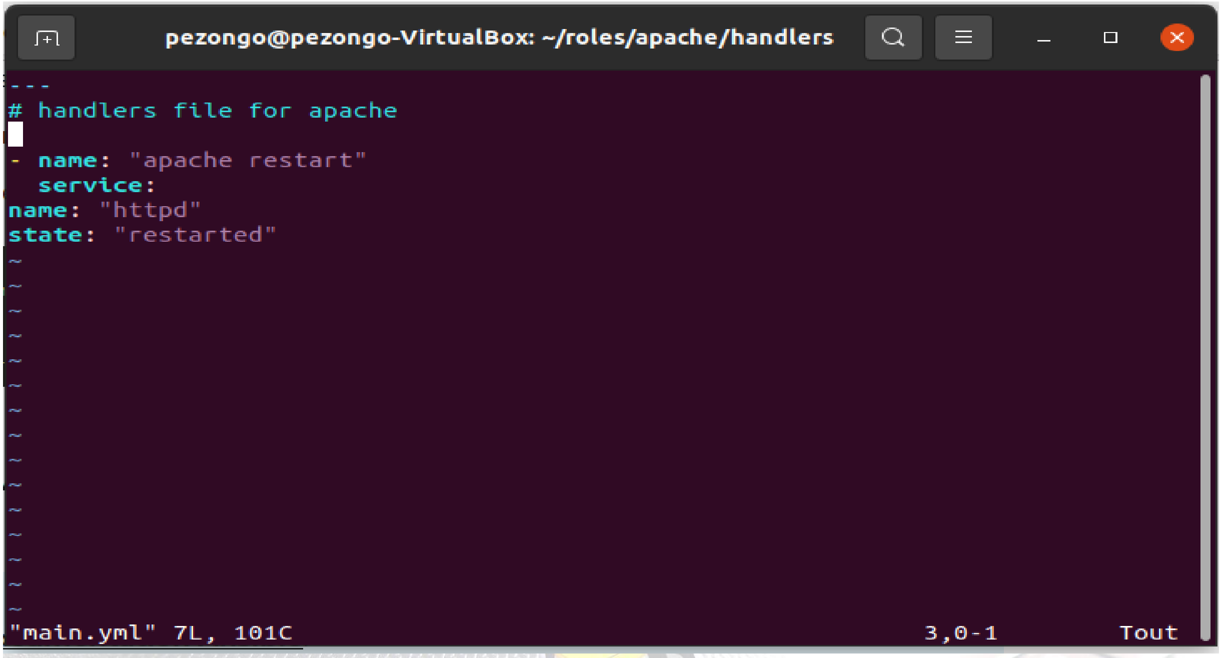
- ➢ The first task, "apache installation" will install the Apache server with the "apt" module. The name: "httpd" indicates the package concerned and the state: "present" specifies that it must be installed.
- ➢ The second task, "apache service activation" will activate the Apache service with the "service" module. The name: "apache2" indicates the concerned service, the state: "started" indicates that the service will be started and the enabled: yes indicates that the service will be activated.
- ➢ The third task, "install php7 packages" includes an external configuration file to install PHP. The task calls with the "include" option to the php7-install.yml file which is placed in the tasks directory next to main.yml. The condition when with the filter (php_install|default(False)|bool) allow to condition the installation of PHP

We will now configure the php installation file in the php7-install.yml directory:



```
---

#installation de php
- name: "installation php"
  apt:
          name: "php7"
          state: "present"
~
~
```

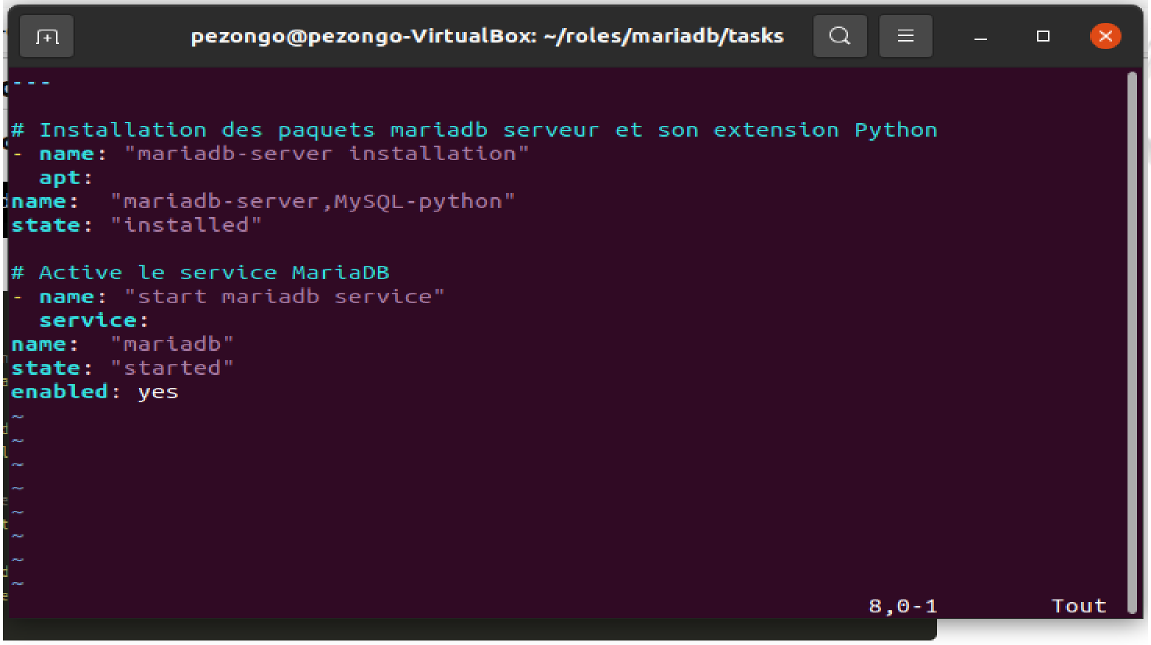
We will use the service module to restart the Apache service by configuring the apache/handlers/main.yml file:



```
---
# handlers file for apache

- name: "apache restart"
  service:
name: "httpd"
state: "restarted"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"main.yml" 7L, 101C                                        3,0-1        Tout
```

Now we will write the different scripts for the different remaining configurations

- In mariadb/tasks/main.yml



```
---

# Installation des paquets mariadb serveur et son extension Python
- name: "mariadb-server installation"
  apt:
name:  "mariadb-server,MySQL-python"
state: "installed"

# Active le service MariaDB
- name: "start mariadb service"
  service:
name:  "mariadb"
state: "started"
enabled: yes
~
~
~
~
~
~
~
~
                                                 8,0-1        Tout
```

VI. PLAYBOOKS

A playbook is a YAML configuration file containing a sequence play, or plays. Each one can be made of options, and calls one or more roles. It is used to describe a deployment or configuration strategy, by structuring the necessary actions.

By using playbooks, we have the possibility to keep the code in a file and to reuse it in your own way, unlike the ansible command which is volatile.

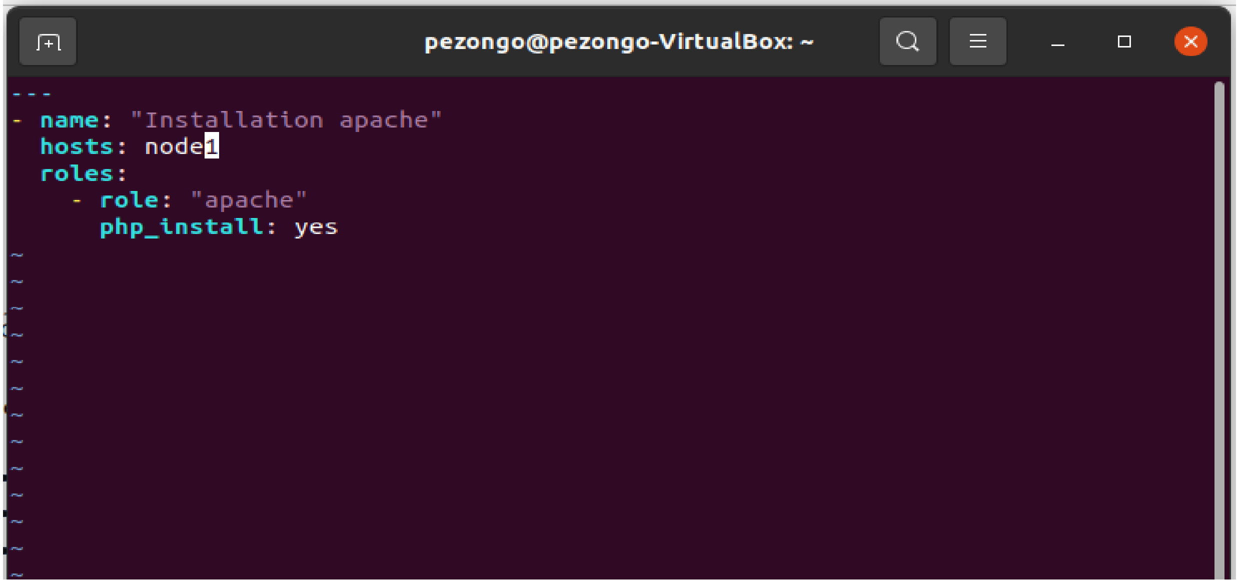
So we will create 2 playbooks:

- One to install Apache.
- One to install MariaDB.

1. For apache installation

The installation of Apache will consist in launching the apache role and in defining the variable which will indicate if PHP must be installed or not.

We create the playbook install-apache.yml to execute the commands.



```
---
- name: "Installation apache"
  hosts: node1
  roles:
    - role: "apache"
      php_install: yes
~
~
~
~
~
~
~
~
~
~
~
~
~
```

- name: "apache installation" indicates the name of the instruction set;

hosts: node1 indicates the concerned node;

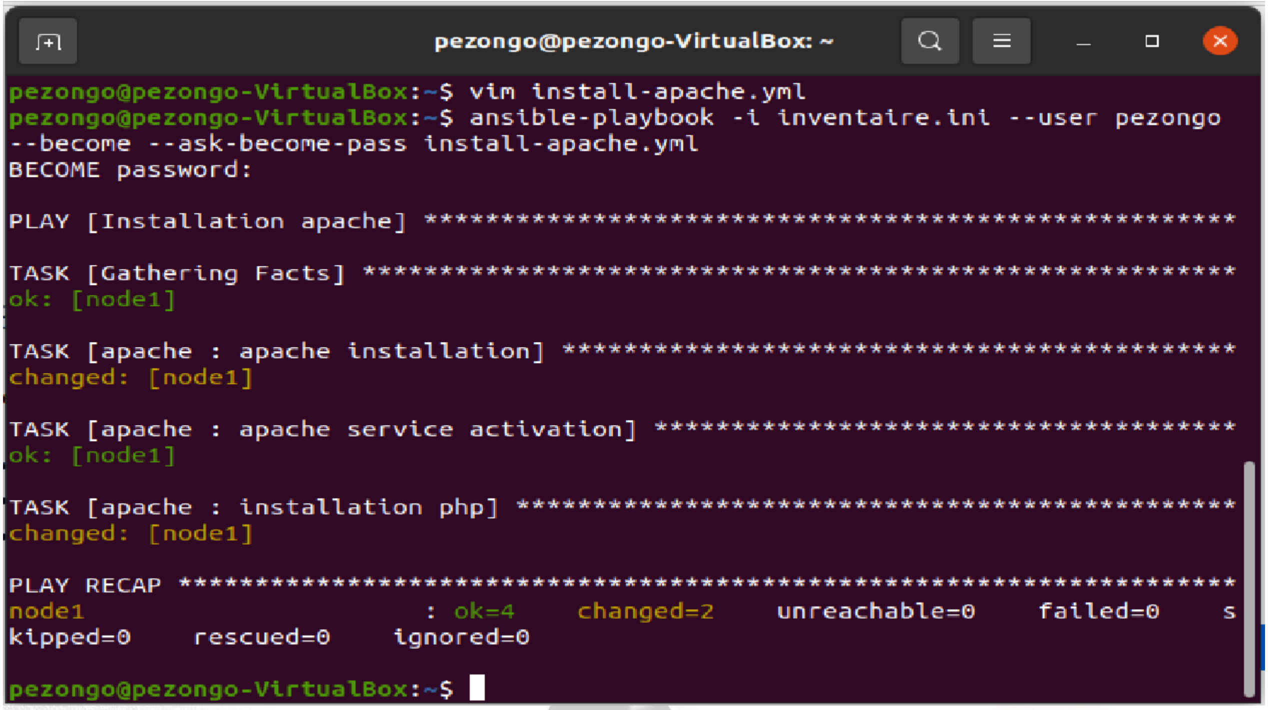
roles: indicates a role section;

- role: "apache" indicates the role to launch;

php_install: yes, indicates the value of the php_install variable

We run the ansible-playbook command to execute the install-apache.yml playbook with the following connection options:

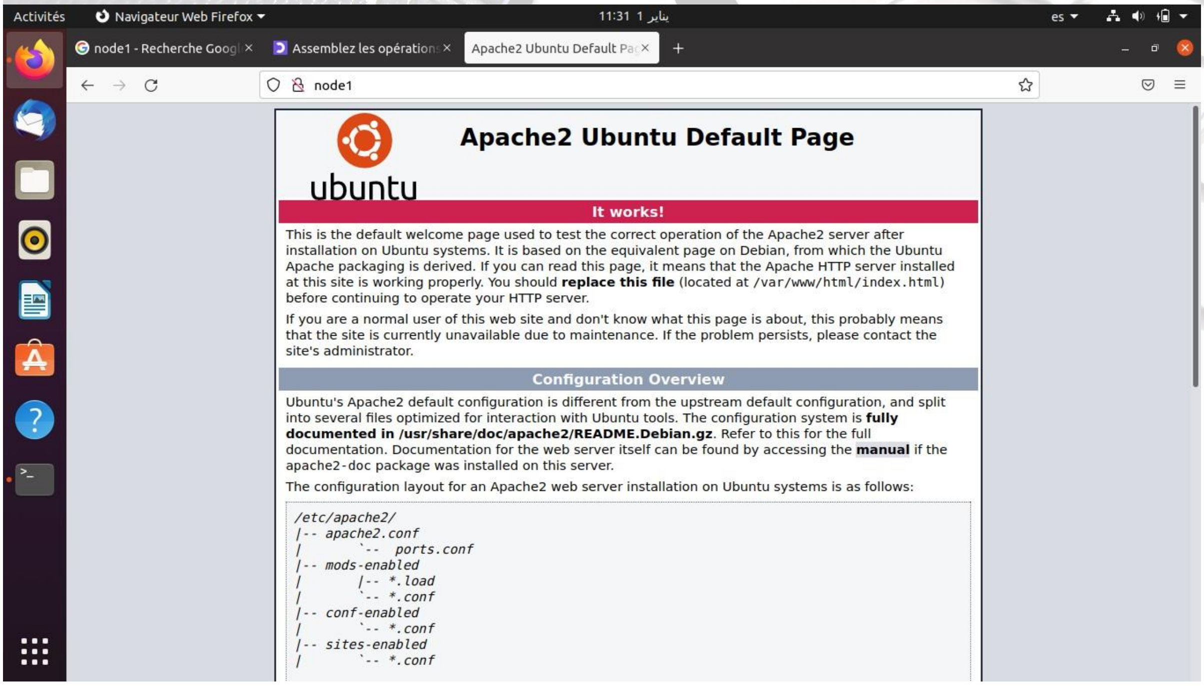
```
ansible-playbook -i inventaire.ini --user pezongo --become --ask-become-pass install-apache.yml
```



```
pezongo@pezongo-VirtualBox:~$ vim install-apache.yml
pezongo@pezongo-VirtualBox:~$ ansible-playbook -i inventaire.ini --user pezongo
--become --ask-become-pass install-apache.yml
BECOME password:

PLAY [Installation apache] *****************************************************

TASK [Gathering Facts] *********************************************************
ok: [node1]

TASK [apache : apache installation] ********************************************
changed: [node1]

TASK [apache : apache service activation] **************************************
ok: [node1]

TASK [apache : installation php] ***********************************************
changed: [node1]

PLAY RECAP *********************************************************************
node1                      : ok=4    changed=2    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

pezongo@pezongo-VirtualBox:~$ 
```

we can check that php is well installed and apache too.

Now we open our browser and we type the address http://node1 to see that apache is well installed on it



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at `/var/www/html/index.html`) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

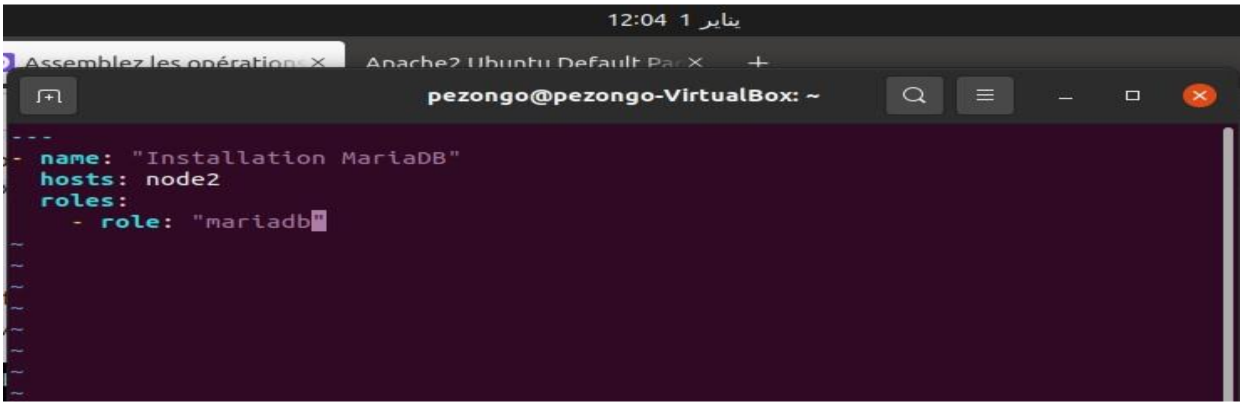
The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

```
/etc/apache2/
|-- apache2.conf
|       `--  ports.conf
|-- mods-enabled
|       |-- *.load
|       `-- *.conf
|-- conf-enabled
|       `-- *.conf
|-- sites-enabled
|       `-- *.conf
```

2. For Mariadb installation

The installation of MariaDB will simply consist in launching the mariadb role.

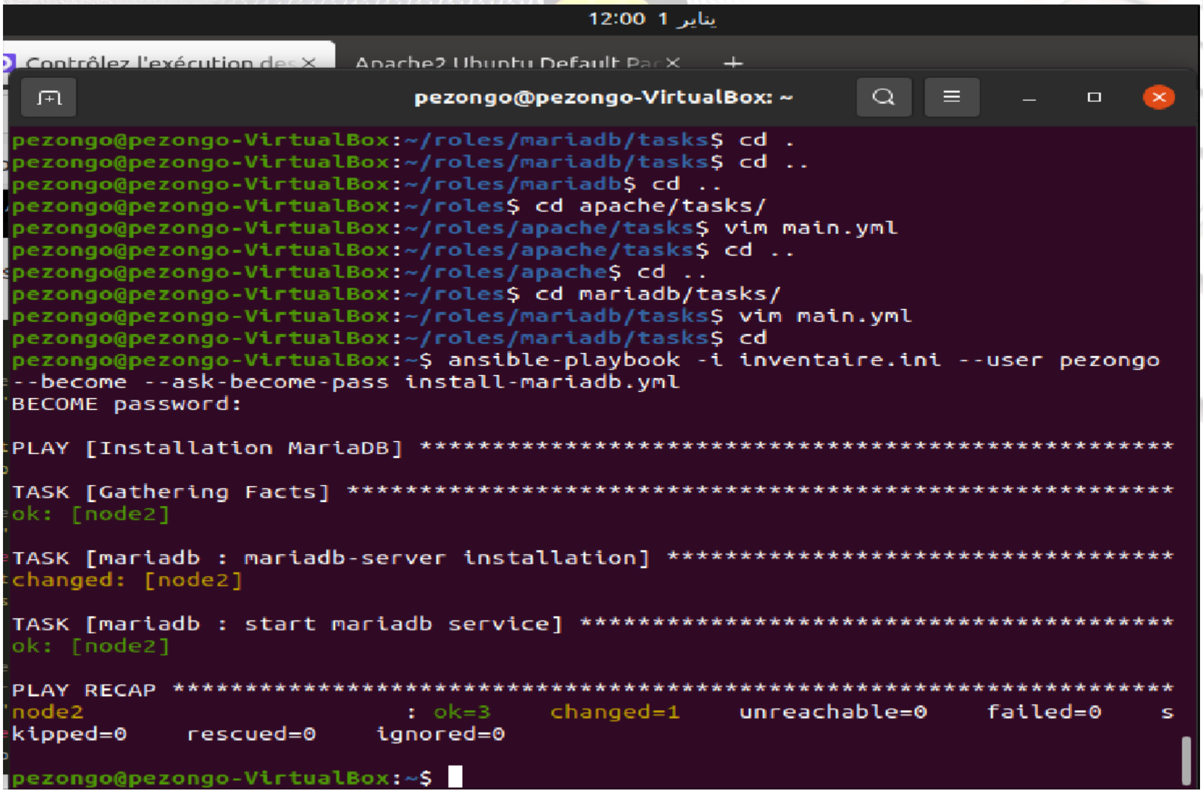
We create the playbook install-mariadb.yml to execute the commands.


12:04 1 يناير
Assemblez les opération ×
Apache2 Ubuntu Default Pa ×
pezongo@pezongo-VirtualBox: ~

```
---
- name: "Installation MariaDB"
  hosts: node2
  roles:
    - role: "mariadb"
~
~
~
~
~
~
~
~
```

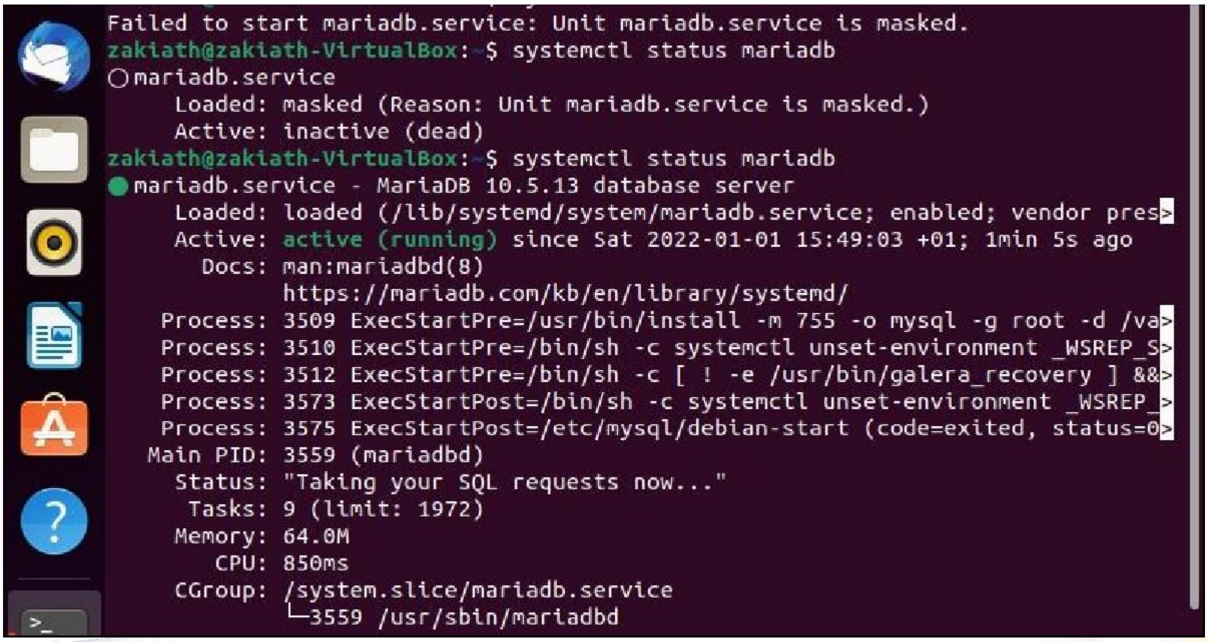
We run the ansible-playbook command to execute the install-mariadb.yml playbook with the following connection options:

```
ansible-playbook -i inventaire.ini --user pezongo --become --ask-become-pass install-mariadb.yml
```

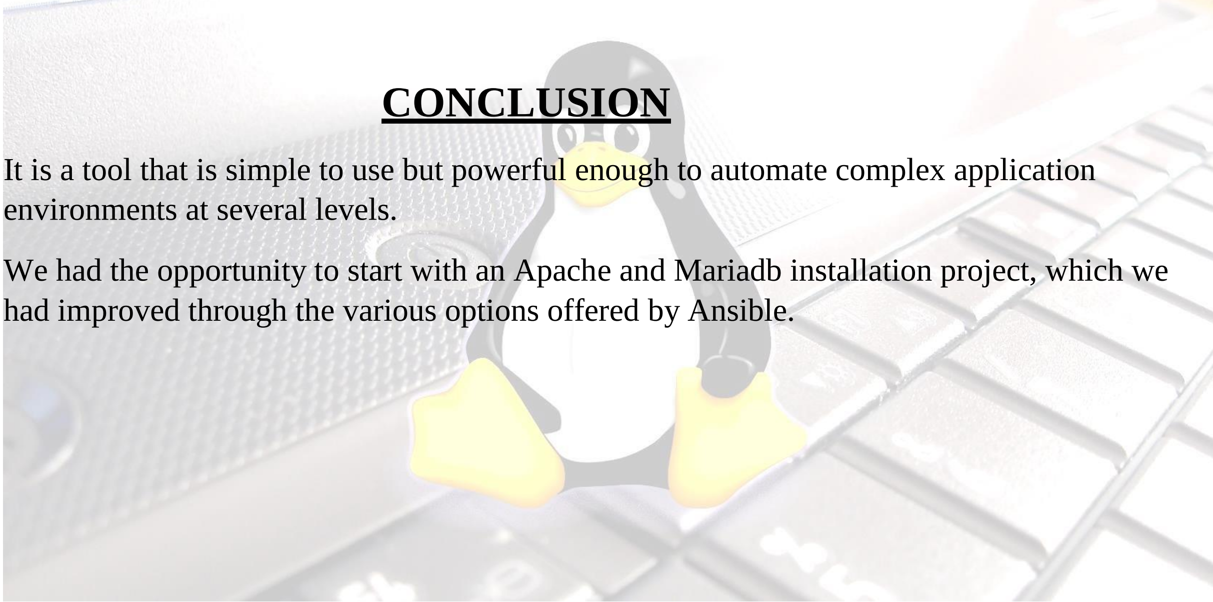

12:00 1 يناير
Contrôlez l'exécution de ×
Apache2 Ubuntu Default Pa ×
pezongo@pezongo-VirtualBox: ~

```
pezongo@pezongo-VirtualBox:~/roles/mariadb/tasks$ cd .
pezongo@pezongo-VirtualBox:~/roles/mariadb/tasks$ cd ..
pezongo@pezongo-VirtualBox:~/roles/mariadb$ cd ..
pezongo@pezongo-VirtualBox:~/roles$ cd apache/tasks/
pezongo@pezongo-VirtualBox:~/roles/apache/tasks$ vim main.yml
pezongo@pezongo-VirtualBox:~/roles/apache/tasks$ cd ..
pezongo@pezongo-VirtualBox:~/roles/apache$ cd ..
pezongo@pezongo-VirtualBox:~/roles$ cd mariadb/tasks/
pezongo@pezongo-VirtualBox:~/roles/mariadb/tasks$ vim main.yml
pezongo@pezongo-VirtualBox:~/roles/mariadb/tasks$ cd
pezongo@pezongo-VirtualBox:~$ ansible-playbook -i inventaire.ini --user pezongo
--become --ask-become-pass install-mariadb.yml
BECOME password:

PLAY [Installation MariaDB] ***************************************************

TASK [Gathering Facts] ********************************************************
ok: [node2]

TASK [mariadb : mariadb-server installation] **********************************
changed: [node2]

TASK [mariadb : start mariadb service] ****************************************
ok: [node2]

PLAY RECAP ********************************************************************
node2                      : ok=3    changed=1    unreachable=0    failed=0    s
kipped=0    rescued=0    ignored=0

pezongo@pezongo-VirtualBox:~$ 
```

we can check that mariadb is well installed



```
Failed to start mariadb.service: Unit mariadb.service is masked.
zakiath@zakiath-VirtualBox:~$ systemctl status mariadb
○ mariadb.service
     Loaded: masked (Reason: Unit mariadb.service is masked.)
     Active: inactive (dead)
zakiath@zakiath-VirtualBox:~$ systemctl status mariadb
● mariadb.service - MariaDB 10.5.13 database server
     Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor pres>
     Active: active (running) since Sat 2022-01-01 15:49:03 +01; 1min 5s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 3509 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /va>
    Process: 3510 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_S>
    Process: 3512 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] &&>
    Process: 3573 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_>
    Process: 3575 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0>
   Main PID: 3559 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 9 (limit: 1972)
     Memory: 64.0M
        CPU: 850ms
     CGroup: /system.slice/mariadb.service
             └─3559 /usr/sbin/mariadbd
```



CONCLUSION

It is a tool that is simple to use but powerful enough to automate complex application environments at several levels.

We had the opportunity to start with an Apache and Mariadb installation project, which we had improved through the various options offered by Ansible.