



Automatisation avec Ansible

Apache et PHP

14 novembre 2024

Table des matières

1	Créer le fichier YAML pour le rôle apache	3
1.1	Écrire le code pour installer Apache	3
1.2	Créer le playbook pour installer Apache et PHP.....	5
1.2.1	La structure d'un playbook.....	7
1.3	Lancer le playbook pour installer Apache sur web1.....	8

1 Créer le fichier YAML pour le rôle apache

1.1 Écrire le code pour installer Apache

Voici le fichier **tasks/main.yml** qui contient les **actions** d'installation d'**Apache** et de **PHP**.

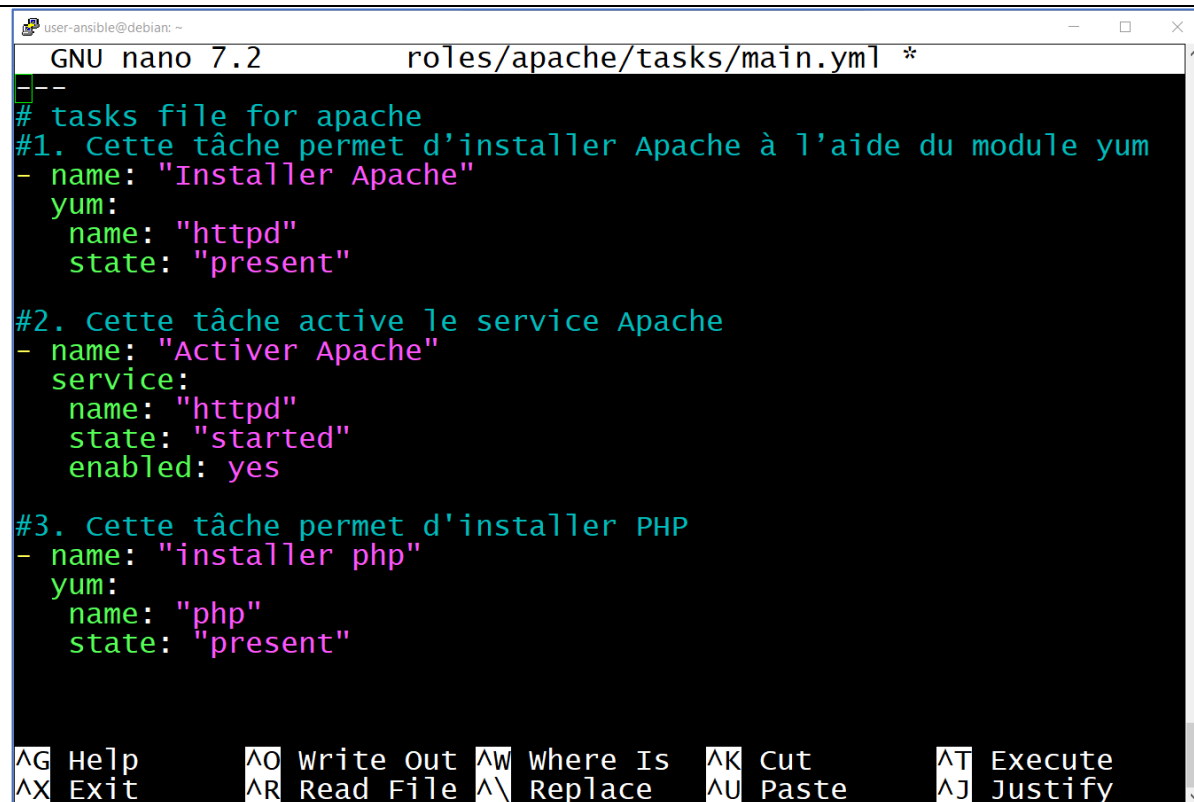
```
(ansible10.5.0) user-ansible@debian:~$ nano roles/apache/tasks/main.yml
```

Le fichier est commenté pour en comprendre le sens.

```
---
# tasks file for apache
#1. Cette tâche permet d'installer Apache à l'aide du module yum
- name: "Installer Apache"
  yum:
    name: "httpd"
    state: "present"

#2. Cette tâche active le service Apache
- name: "Activer Apache"
  service:
    name: "httpd"
    state: "started"
    enabled: yes

#3. Cette tâche permet d'installer PHP
- name: "installer php"
  yum:
    name: "php"
    state: "present"
```



```
GNU nano 7.2 roles/apache/tasks/main.yml *
---
# tasks file for apache
#1. Cette tâche permet d'installer Apache à l'aide du module yum
- name: "Installer Apache"
  yum:
    name: "httpd"
    state: "present"

#2. Cette tâche active le service Apache
- name: "Activer Apache"
  service:
    name: "httpd"
    state: "started"
    enabled: yes

#3. Cette tâche permet d'installer PHP
- name: "installer php"
  yum:
    name: "php"
    state: "present"
```

On va voir en détail chacune des tâches décrites dans ce fichier :

1) La **première tâche**, "**Installer Apache**" va installer le service Apache avec le module **yum**.

- Le champ **name**: "**httpd**" indique le nom du paquetage.
- Le champ **state**: "**present**" spécifie qu'il faut l'installer.

2) La **deuxième tâche**, "**Activer Apache**" va activer le service Apache avec le module **service**.

- Le champ **name**: "**httpd**" indique le service concerné.
- Le champ **state**: "**started**" indique que le service sera démarré.
- Le champ **enabled**: **yes** indique que le service sera activé.

3) La **troisième tâche**, "**Installer php**" va installer le PHP avec le module **yum**.

- Le champ **name**: "**php**" indique le nom du paquetage.
- Le champ **state**: "**present**" spécifie qu'il faut l'installer.

L'indentation est importante. Il faut décaler de deux espaces chaque ligne pour respecter l'alignement logique de chaque tâche. Si on utilise un éditeur de code de type Visual Studio Code, l'indentation sera automatique en sélectionnant la syntaxe YAML dans l'éditeur.

1.2 Créer le playbook pour installer Apache et PHP

On a contrôlé l'exécution des opérations et enchaîné plusieurs actions, en écrivant du code Ansible dans les fichiers de configuration présents dans le rôle **apache**.

À présent, on va assembler toutes ces opérations et automatiser l'installation d'Apache. Pour cela, on va utiliser les **playbooks** Ansible.

Ensuite, on jouera ces playbooks avec la commande `ansible-playbook`, afin de déployer automatiquement MediaWiki.

Un playbook est un **fichier de configuration YAML** contenant une suite de jeux d'instructions, ou *plays* en anglais. Chacun peut être constitué d'options, et fait appel à **un ou plusieurs rôles**. Il permet de décrire une **stratégie de déploiement**, ou de configuration, en **structurant** les actions nécessaires.

En utilisant les **playbooks**, on a la possibilité de conserver le code dans un fichier et de le réutiliser à notre façon, contrairement à la commande **ansible** qui est **volatile**.

On va créer un playbook qui nous permettra d'**installer Apache**.

L'ordre d'exécution d'un playbook est important, car les rôles de configuration dépendent des rôles d'installation.

Les services **Apache** et **MariaDB** doivent être **installés avant** que la **configuration** de MediaWiki puisse être lancée.

Toutes les **opérations** qui vont suivre sont à faire sur le contrôleur Ansible (**node manager**).

Tout d'abord, Se connecter sur le **node manager** :

```
root@debian:~# su - user-ansible
user-ansible@debian:~$
```

Puis, activer l'**environnement virtuel**:

```
user-ansible@debian:~$ source ansible10.5.0/bin/activate
(ansible10.5.0) user-ansible@debian:~$
```

L'installation d'Apache et de PHP va consister à lancer le rôle **apache**.

Créer le playbook **install-apache.yml** à la racine de l'environnement virtuel :

```
(ansible10.5.0) user-ansible@debian:~$ nano install-apache.yml
```

Le contenu du fichier ressemble à celui-ci :

```
---  
- name: "Installation apache"  
  hosts: web1  
  roles:  
    - role: "apache"
```

```
---  
- name: "Installation apache"  
  hosts: web1  
  roles:  
    - role: "apache"
```

1.2.1 La structure d'un playbook

Le playbook est un fichier **YAML**. Il est structuré de la façon suivante :

- il commence par **3 tirets** ;
- ensuite, il peut y avoir un **bloc général**, constitué d'un **en-tête** composé du **nom des nodes ou du groupe concerné**, de **variables**, d'**options**, etc.
- puis, on trouve des **blocs spécifiques** qui définissent les **jeux d'instruction** (ou *play* en anglais). Chaque jeu d'instruction est composé du **nom du jeu** (avec un tiret au début), et de façon **optionnelle**, d'un **en-tête** et d'une **section** qui peut prendre plusieurs formes comme des **tasks**, des **rôles**, des **handlers**... en fonction de l'action demandée.

Dans le cas du playbook `install-apache.yml` ci-dessus, il y a 3 jeux d'instruction.

- Chaque **ligne** à l'intérieur d'un bloc est **indentée**, et chaque début de ligne est décalé de **deux espaces**.

Comme on peut le voir, la principale différence entre un **playbook** et un fichier de configuration de type `main.yml` est que le **playbook** contient une liste de jeux d'instructions (**plays**) et que le `main.yml` contient une liste de tâches (**tasks**) ou de **variables**.

On trouve un descriptif plus détaillé du [fonctionnement des playbooks sur sa documentation](#).

Dans `install-apache.yml`, il n'y a pas de bloc général. Par contre, il y a un bloc spécifique pour définir le jeu d'instructions qui consiste à lancer le rôle apache. Il y a également un en-tête qui indique sur quel node il faut lancer les actions.

Le jeu d'instructions est défini de la façon suivante :

- - name: "Installation apache" indique le nom du jeu d'instructions ;
- hosts: web1 indique le node concerné ;
- roles: indique une section rôles ;
- - role: "apache" indique le rôle à lancer ;

1.3 Lancer le playbook pour installer Apache sur web1

Lancer la commande `ansible-playbook` pour exécuter le playbook `install-apache.yml` avec les **options** de connexion suivantes :

```
(ansible10.5.0) user-ansible@debian:~$ ansible-playbook -i inventaire.ini --  
user user-ansible --become --ask-become-pass install-apache.yml  
  
BECOME password:  
  
PLAY [Installation apache] *****  
  
TASK [Gathering Facts] *****  
[WARNING]: Platform linux on host web1 is using the discovered Python  
interpreter at /usr/bin/python3.12, but future installation of another Python  
interpreter could change the meaning of that path. See  
https://docs.ansible.com/ansible-  
core/2.17/reference\_appendices/interpreter\_discovery.html for more information.  
ok: [web1]  
  
TASK [apache : Installer Apache] *****  
changed: [web1]  
  
TASK [apache : Activer Apache] *****  
changed: [web1]  
  
TASK [apache : installer php] *****  
changed: [web1]  
  
PLAY RECAP *****  
web1 : ok=4 changed=3 unreachable=0 failed=0 s kipped=0 rescued=0 ignored=0  
  
(ansible10.5.0) user-ansible@debian:~$
```

On a utilisé la commande `ansible-playbook` avec les mêmes options que celles utilisées avec la commande `ansible`. Seul le **node concerné** est directement défini dans le **playbook**.

La commande `ansible-playbook` fait partie des **outils installés** avec Ansible. Cette commande permet de lancer des playbooks. Les options sont similaires à celles de la commande `ansible` utilisée précédemment.

Le résultat (appelé **callback d'affichage**) prend la forme suivante :

```
(ansible10.5.0) user-ansible@debian:~$ ansible-playbook -i inventaire.ini --user user-ansible --become --ask-become-pass install-apache.yml
BECOME password:

PLAY [Installation apache] *****

TASK [Gathering Facts] *****
[WARNING]: Platform linux on host web1 is using the discovered python interpreter at /usr/bin/python3.12, but future installation of another Python interpreter could change the meaning of that path. See https://docs.ansible.com/ansible-core/2.17/reference_appendices/interpreter_discovery.html for more information.
ok: [web1]

TASK [apache : Installer Apache] *****
changed: [web1]

TASK [apache : Activer Apache] *****
changed: [web1]

TASK [apache : installer php] *****
changed: [web1]

PLAY RECAP *****
web1 : ok=4 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

(ansible10.5.0) user-ansible@debian:~$
```

On peut constater que le **callback** donne la **liste des tâches** pour **chaque action** définie dans le rôle **apache**, avec leur **état** (changed). Ce qui signifie que toutes les actions ont provoqué un changement sur le node **web1**.

La **dernière ligne** du callback indique le **récapitulatif** par node, de l'exécution des **tâches** selon les quatre états possibles (**ok**, **changed**, **unreachable**, **failed**).

Il est temps de vérifier sur le node **web1**, si **Apache** et **PHP** ont bien été installés.

Se connecter sur le node **web1** et lancer la commande qui permet de connaître la **version de PHP** :

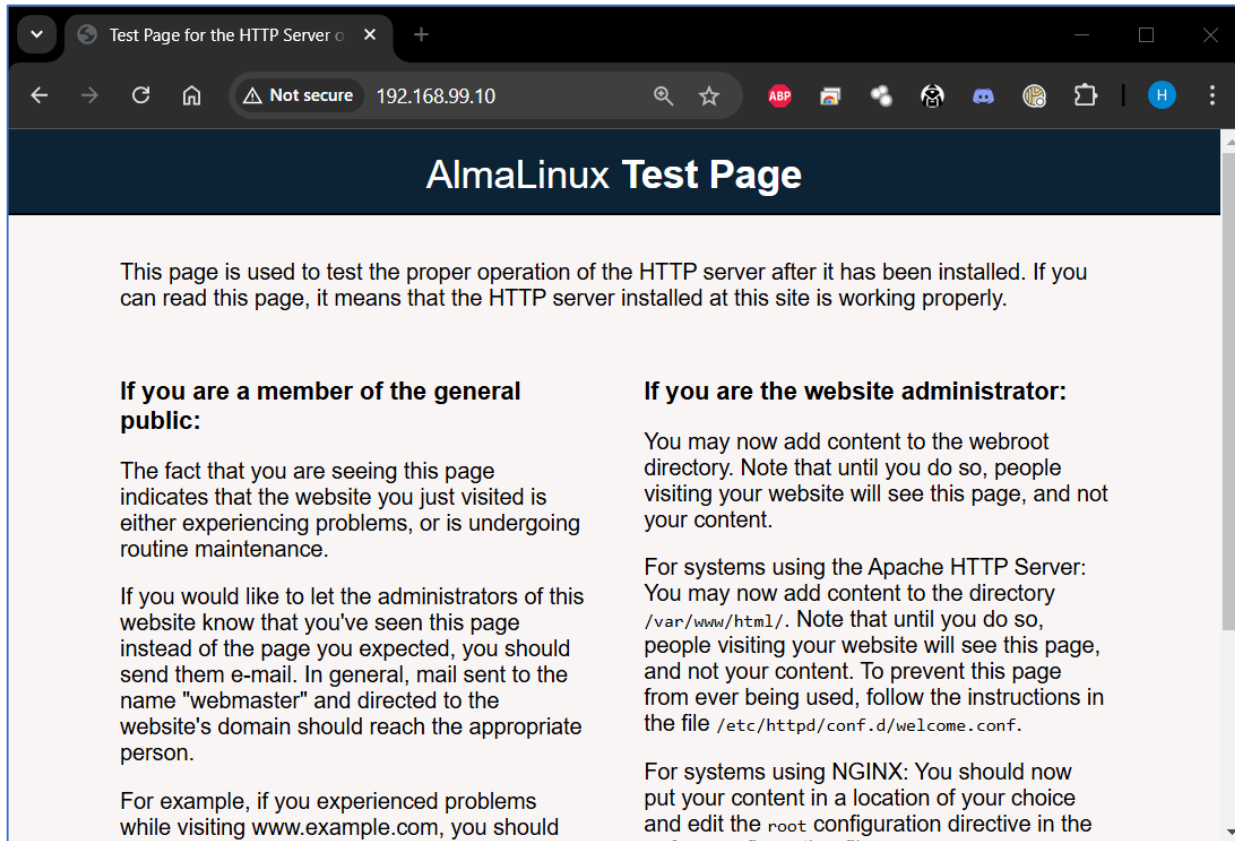
```
(ansible10.5.0) user-ansible@debian:~$ ssh user-ansible@web1
Last login: Thu Nov  7 09:02:57 2024 from 192.168.99.130
```

```
[user-ansible@web1 ~]$ php --version
PHP 8.0.30 (cli) (built: Aug  3 2023 17:13:08) ( NTS gcc x86_64 )
Copyright (c) The PHP Group
Zend Engine v4.0.30, Copyright (c) Zend Technologies
    with Zend OPcache v8.0.30, Copyright (c), by Zend Technologies
```

PHP est bien installé.

Lancer un navigateur avec l'URL <http://192.168.99.10>

192.168.99.10 étant l'adresse IP de web1.



Par défaut, Apache affiche une page de test, ce qui permet de vérifier que le service est opérationnel.