Automatisation avec Ansible

Contrôler l'exécution des opérations avec yaml

7 novembre 2024

Table des matières

1 Introduction

Dans le module précédent, on a organisé le déploiement de MediaWiki en créant des rôles structurés. On a transposé les 6 étapes nécessaires au déploiement de MediaWiki en opérations d'installation et de configuration. Ensuite, on a créé 5 rôles pour intégrer ces opérations dans une structure arborescente.

Dans ce module, on va construire les scripts d'automatisation en complétant les fichiers **main.yml** présents dans chaque rôle. Ce qui permettra d'exécuter des **tâches** et d'enchaîner plusieurs **actions**. Pour nous aider à construire les scripts, on suivra les **6 étapes** d'installation de MediaWiki qui sont détaillées dans le **guide d'installation** MediaWiki.

Reprendre l'arborescence des rôles construite précédemment :

```
(ansible10.5.0) user-ansible@debian:~$ tree roles/
roles/
├── apache
│   ├── defaults
│   │   └── main.yml
│   ├── files
│   ├── handlers
│   │   └── main.yml
│   ├── meta
│   │   └── main.yml
│   ├── README.md
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   ├── tests
│   │   ├── inventory
│   │   └── test.yml
│   └── vars
│       └── main.yml
├── commun
│   ├── defaults
│   │   └── main.yml
│   ├── files
│   ├── handlers
│   │   └── main.yml
│   ├── meta
│   │   └── main.yml
│   ├── README.md
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   ├── tests
│   │   ├── inventory
│   │   └── test.yml
│   └── vars
│       └── main.yml
├── confapache
│   ├── defaults
│   │   └── main.yml
│   ├── files
│   ├── handlers
│   │   └── main.yml
```

```
│   ├── meta
│   │   └── main.yml
│   ├── README.md
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   ├── tests
│   │   ├── inventory
│   │   └── test.yml
│   └── vars
│       └── main.yml
├── confdb
│   ├── defaults
│   │   └── main.yml
│   ├── files
│   ├── handlers
│   │   └── main.yml
│   ├── meta
│   │   └── main.yml
│   ├── README.md
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   ├── tests
│   │   ├── inventory
│   │   └── test.yml
│   └── vars
│       └── main.yml
├── mariadb
│   ├── defaults
│   │   └── main.yml
│   ├── files
│   ├── handlers
│   │   └── main.yml
│   ├── meta
│   │   └── main.yml
│   ├── README.md
│   ├── tasks
│   │   └── main.yml
│   ├── templates
│   ├── tests
│   │   ├── inventory
│   │   └── test.yml
│   └── vars
│       └── main.yml
└── mediawiki
    ├── defaults
    │   └── main.yml
    ├── files
    ├── handlers
    │   └── main.yml
    ├── meta
    │   └── main.yml
    ├── README.md
    ├── tasks
    │   └── main.yml
    ├── templates
    ├── tests
    │   ├── inventory
```

```
    │   └── test.yml
    └── vars
        └── main.yml

55 directories, 48 files
(ansible10.5.0) user-ansible@debian:~$
```

On distingue, dans cette **arborescence**, **6 rôles** correspondant à **9 fichiers YAML** décrivant les **rôles**:

- apache
- commun
- confapache
- confdb
- mariadb
- mediawiki

On va donc créer **les fichiers YAML** pour les **6 rôles**:

1) Le rôle **apache**

   - Un fichier **tasks/main.yml** contient les actions pour installer Apache et PHP et activer Apache;

2) Le rôle **commun** de MediaWiki

   - Un fichier **defaults/main.yml** contient les variables d'installation qui seront utilisées dans les rôles suivants.

3) Le rôle **confapache** de MediaWiki :

   - Un fichier **meta/main.yml** contient la dépendance avec le rôle commun ;
   - Un fichier **tasks/main.yml** contient les actions pour configurer Apache pour MediaWiki.

4) Le rôle **confdb** de MediaWiki :

   - Un fichier **meta/main.yml** contient la dépendance avec le rôle commun ;
   - Un fichier **tasks/main.yml** contient les actions pour configurer MariaDB pour MediaWiki.

5) Le rôle **mariadb**

   - Un fichier **tasks/main.yml** contient les actions pour installer MariaDB.

Il est possible de créer dans le répertoire **tasks** autant de fichiers de configuration qu'on veut.

Les rôles peuvent être indépendants ou dépendants les uns des autres. Par exemple, les rôles **apache**, **mariadb** et **commun** sont indépendants et peuvent être utilisés séparément. Par contre, les rôles **confapache** et **confdb** dépendent du rôle **commun**.

On va maintenant **compléter** chaque fichier **de configuration YAML**.

2 Construire vos fichiers YAML

Un **fichier de configuration YAML** contenu dans les **rôles** peut contenir une **liste de tâches** ou une **liste de variables**. YAML est l'acronyme de **Yet Another Markup Language**

- Les fichiers YAML commencent toujours par **3 tirets** (---).
- Ensuite, on a les différentes **tâches** successives qui commencent par **1 tiret** (-) et le **nom de la tâche**.
- Chaque tâche utilise un **module** avec ses **arguments** ou ses **options**.
- Les **arguments** ou les **options** sont décalés à la ligne de **2 espaces**.
- Pour construire le fichier, On doit chercher dans la **documentation Ansible** quel **module** utiliser.

Pour installer Apache, il faut utiliser un gestionnaire de paquetages dnf qui a un module Ansible permettant d'automatiser l'installation des paquetages.

Si on veut en savoir plus sur l'utilisation d'un module Ansible, comme par exemple pour le module **dnf** ou **yum**, on peut utiliser la commande suivante :

```
(ansible10.5.0) user-ansible@debian:~$ ansible-doc dnf
> MODULE ansible.builtin.dnf (/home/user-ansible/ansible10.5.0/lib/python3.11/site-
packages/a>

  Installs, upgrade, removes, and lists packages and groups with the
  dnf package manager.

  * note: This module has a corresponding action plugin.

OPTIONS (red indicates it is required):

   allow_downgrade  Specify if the named package and version is allowed to downgrade a maybe
                    already installed higher version of that package. Note
                    that setting allow_downgrade=True can make this module
                    behave in a non-idempotent way. The task could end up
                    with a set of packages that does not match the complete
                    list of specified packages to install (because
                    dependencies between the downgraded package and others
                    can cause changes to the packages which were in the
                    earlier transaction).
        default: 'no'
        type: bool

   allowerasing  If `true' it allows  erasing  of  installed  packages to resolve
                 dependencies.
        default: 'no'
        type: bool

   autoremove  If `true', removes all "leaf" packages from the system that were
               originally installed as dependencies of user-installed
               packages but which are no longer required by any such
:
```