CPE Lyon - 4ETI, 3IRC & 3ICS - Année 2022/2023
Administration Système
TP 8 - Ansible

Dans ce TP, vous allez découvrir et prendre en main le logiciel *Ansible*, qui permet d'automatiser la configuration, de déployer des applications et d'orchestrer des tâches avancées sur des machines, telles des déploiements continus. Ne soyez pas effrayés par la longueur du sujet, il y a beaucoup plus à lire qu'à faire !

L'objectif de ce TP est de déployer un wiki sur des serveurs afin de partager de la documentation (informations, bonnes pratiques de développement, etc.). Mais plutôt que de procéder aux différentes installations à la main, vous allez créer et utiliser des scripts pour *automatiser* ces installations sur les différents serveurs.

Infrastructure-As-Code

Cette manière de procéder constitue ce qu'on appelle l'***Infrastructure as Code***; c'est un concept qui permet de gérer les tâches d'administration automatiquement, via du code, à l'aide de fichiers de définition plutôt que faire des traitements manuels (répétitifs ou rébarbatifs...). Ainsi, vous pourrez réutiliser ces scripts pour un autre déploiement ; vous gagnerez du temps et vous pourrez également partager les scripts avec vos collègues.

Quel outil utiliser ?

Il faut que vous trouviez un ***outil de gestion de configuration***. Il en existe plusieurs, mais les plus connus et les plus utilisés sont **Chef**, **Puppet** et **Ansible**.
Ils ont tous des avantages et des inconvénients ; le choix doit porter sur :

- le **contexte de l'environnement technique** (un outil est déjà utilisé en interne, le nombre de serveurs à gérer est important...) ;
- les **connaissances internes** des équipes techniques : Chef et Puppet utilisent **Ruby**, alors que Ansible utilise **Python**. **Ansible est simple à utiliser** ; les autres outils, un peu moins, le temps d'appropriation est donc un peu plus long ;
- le **besoin** : est-ce que vous voulez faire de la conformité, du déploiement, de la performance, de la simplicité, de l'intégration, de la sécurité... ?

Pour ces raisons, le choix s'est porté pour ce TP sur **Ansible**.

Ansible a été initié en 2012 par **Michael DeHaan**, puis acquis par Red Hat en 2015. Red Hat ayant été racheté par IBM en 2018, Ansible est désormais la propriété d'IBM. Le nom *Ansible* a quant à lui été choisi en référence au terme Ansible utilisé par **Ursula Le Guin** dans ses romans de science-fiction pour désigner un moyen de communication plus rapide que la lumière.

Références

- Documentation officielle d'Ansible
- Cours OpenClassrooms (dont est largement inspiré ce TP !)
- Vidéos Hackademy, très simples et bien faites
- Bonne présentation d'Ansible : https://www.youtube.com/watch?v=jePp5ZP1n14
- Vidéo d'introduction à Ansible par Grafikart : https://www.youtube.com/watch?v=DwNapBHypE8

Depuis la version 3.0, Ansible est distribué en deux paquets (python) afin de faciliter le développement et des releases plus rapides des modules, indépendamment du cœur d'Ansible :

- `ansible-core` (nommé `ansible-base` dans la version 2.10), qui contient les programmes de base : **`ansible-playbook`**, **`ansible-galaxy`**, **`ansible-doc`**…, ainsi que quelques modules et de la documentation ;
- le reste des modules et plugins a été déplacé dans diverses *collections* dont la plus utilisée est la `community.general` (voir plus bas).

Version d'Ansible	Version d'ansible-core	Date de sortie	Notes
Ansible 2.9		10/2019	
Ansible 3	`ansible-base` 2.10	02/2021	Nouvelle numérotation
Ansible 4	`ansible-core` 2.11	05/2021	`ansible-base` est renommé `ansible-core`
Ansible 5	`ansible-core` 2.12	11/2021	
Ansible 6	`ansible-core` 2.13	06/2022	

Ansible est un système "sans agent" (*agentless*) : contrairement à des logiciels similaires comme `Puppet`, `Chef` ou `SaltStack`, on n'a pas besoin d'installer un logiciel spécifique sur les machines contrôlées, **tout se passe par `SSH` et `SFTP` depuis la machine de supervision**.

Exercice 1. Mise en place de l'architecture technique

MediaWiki

Il existe différentes solutions de wiki. Nous optons ici pour `MediaWiki` qui est l'une des plateformes les plus connues (c'est notamment sur elle que repose *Wikipédia*). Le guide d'installation propose les étapes suivantes :

1. Installer un **serveur web** pour servir les pages à un navigateur web ;
2. Installer **PHP** pour exécuter le logiciel ;
3. Installer une **base de données** pour stocker les pages ;
4. Télécharger les **fichiers sources** de MediaWiki et les mettre sur le serveur web ;
5. Configurer le serveur web pour pointer vers l'**URL MediaWiki** ;
6. Finaliser l'installation de MediaWiki via le **script d'installation**.

Une architecture modulaire

Nous faisons le choix d'installer un serveur Linux (**Ubuntu**) avec un service web (**Apache**, **PHP**) et un serveur Linux (**Ubuntu**) avec une base de données (**MariaDB**) (Figure 1).

Pourquoi deux serveurs ?

Pourquoi séparer les services sur deux serveurs, alors qu'un seul aurait pu suffire ?
Dans une optique d'extensibilité (ou "scalabilité"), il est toujours intéressant de séparer les services et d'intégrer une architecture modulaire dès la conception. Ainsi, il sera très facile pour faire évoluer l'infrastructure, d'ajouter un serveur supplémentaire. Par exemple, si un seul serveur web n'est pas suffisant pour absorber tout le trafic web, alors il sera possible d'ajouter un serveur web supplémentaire. Avec Ansible, c'est extrêmement simple.

Evidemment, l'installation de ces serveurs (web et bases de données) sera *automatisée*. Pour cela, nous utiliserons une troisième machine, dite *de supervision*, appelée **node manager** (ou **control node**) dans la terminologie Ansible ; les deux machines supervisées, quant à elles, sont appelées des **nodes** (ou **managed nodes** ou **hosts**). Comme indiqué plus haut, le node manager communique avec les nodes par SSH. L'architecture finale à mettre en place est donc celle de la figure 2.

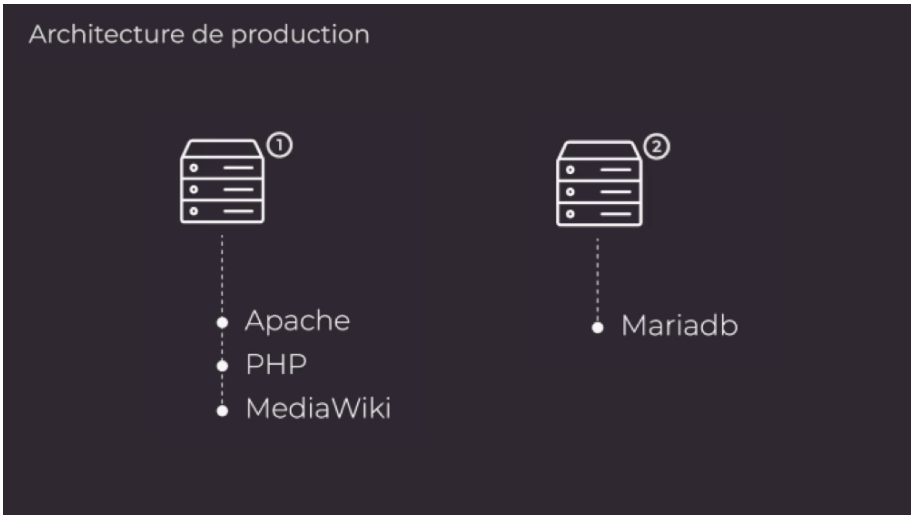

FIGURE 1 – Logiciels à installer sur chaque serveur (© OpenClassrooms)

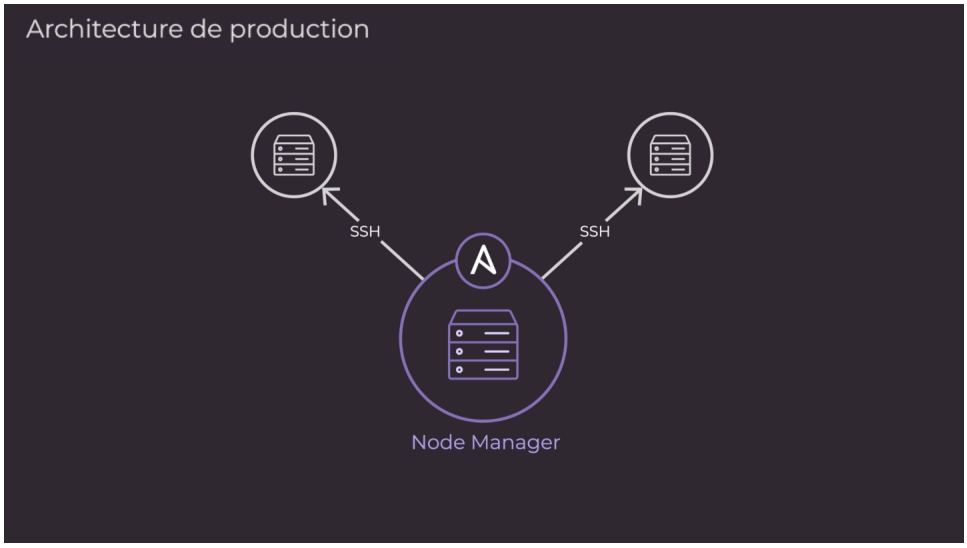

FIGURE 2 – Architecture à mettre en place (© OpenClassrooms)

Question 1. Préparation des machines
Pour ce TP, vous aurez donc besoin de trois machines virtuelles Ubuntu, toutes connectées à Internet, et qui communiqueront entre elles sur le réseau `192.168.122.0/24`, avec les adresses IP suivantes :

- **node manager** : 192.168.122.10
- **serveur 1 (web)** : 192.168.122.11
- **serveur 2 (base de données)** : 192.168.122.12

Afin d'éviter d'effectuer le même travail plusieurs fois, nous allons d'abord configurer le Node Manager entièrement, puis nous en ferons deux copies.

1. **Dans les paramètres de VirtualBox (menu Fichier > Paramètres > Réseau), commencez par créer le réseau NAT correspondant** ;
2. **ajoutez une deuxième interface réseau à la machine, en mode *Réseau NAT* ;**

3. **démarrez la machine, puis modifiez le fichier de configuration de Netplan pour attribuer l'IP statique indiquée ci-dessus**.

Toutes les informations nécessaires se trouvent dans les cours et TP précédents !

Question 2. Résolution des noms
Pour que le node manager communique avec les serveurs par un nom plutôt que par une adresse IP, il faudrait configurer un serveur DNS comme nous l'avons fait dans le TP 6. Cependant, il s'agit d'un processus assez long, et l'objet du présent TP étant Ansible, nous allons utiliser une méthode plus rapide.

Ajoutez les deux entrées suivantes au fichier `/etc/hosts` :

- `192.168.122.11 http1`
- `192.168.122.12 bdd1`

Le fichier **`/etc/hosts`** est en quelques sortes l'ancêtre des serveurs DNS, à une époque où la résolution de noms se faisait à la main... Il est toujours utilisé, et est consulté avant de faire appel à un serveur DNS.

Vous pouvez à présent créer deux clones de cette machine.

Question 3. Test de la configuration

1. **modifiez l'adresse IP de chaque serveur** ;
2. **renommez le serveur web en `http1` et le serveur de base de données en `bdd1`** (reconnectez-vous pour que les modifications apparaissent) ;
3. **Vérifiez que les trois machines arrivent à communiquer entre elles par leur nom et à aller sur Internet**.

Pensez à effectuer des redirections de ports pour vous connecter à vos machines depuis votre machine hôte et profiter d'un terminal plus confortable. Le port 22 (SSH) de chaque machine doit être redirigé, par exemple vers les ports 2222, 2223 et 2224 de la machine hôte.

Vous devez disposer sur chaque machine d'un utilisateur capable d'exécuter des commandes **`sudo`** (il s'agit de l'utilisateur **`tp`** si vous utilisez la machine fournie lors du premier TP) ; dans la suite, nous l'appellerons ***`admin`***.

Exercice 2. Installation et configuration d'Ansible

Question 4. Installation
Il existe plusieurs manières d'installer Ansible :

- *via* les **packages logiciels** sur un système Linux,
- *via* le gestionnaire **pip** de **Python**,
- *via* les **sources officielles** (archives ou Git) maintenues par Red Hat.

L'installation d'un paquet est probablement la solution la plus simple, mais il est fort probable que le dépôt officiel de votre distribution ne contienne pas la dernière version d'Ansible. Nous utiliserons donc un **PPA**, ainsi que le suggère la documentation d'Ansible :

```
$ sudo apt update
$ sudo apt install software-properties-common
$ sudo add-apt-repository --yes --update ppa:ansible/ansible
$ sudo apt install ansible
```

Le paquet `software-properties-common` fournit la commande `add-apt-repository` qui permet d'éviter d'éditer à la main le fichier `sources.list.d` comme nous le faisions dans le TP 4.

Confirmation d'installation

Si tout s'est bien déroulé, vous devriez constater la présence de **10 outils** dans le dossier `/usr/bin` dont le nom commence par `ansible`, parmi lesquels :

- `ansible` : permet de lancer des actions Ansible en mode **ad-hoc** (i.e. en ligne de commande) ;
- `ansible-config` : permet de gérer la configuration de Ansible (ou de la visualiser avec la commande **ansible-config list**) ;
- `ansible-doc` : permet d'obtenir de l'aide pour utiliser Ansible.

Question 5. Utilisateur `user-ansible` sur le node manager
Il est toujours déconseillé de travailler avec un compte **root** (c'est d'ailleurs pour cela que ce compte est désactivé par défaut sous Ubuntu). Il est en revanche conseillé d'utiliser un compte *dédié à Ansible* : il sera ainsi beaucoup plus facile de tracer les actions dans les logs pour débuguer.

Sur le *node manager*, créez un utilisateur `user-ansible` et ajoutez-le au groupe `sudo`.

Le groupe `sudo` est le groupe des utilisateurs ayant l'autorisation d'exécuter n'importe quelle commande, ainsi que mentionné dans le fichier `/etc/sudoers` :

```
# Allow members of group sudo to execute any command
%sudo   ALL=(ALL:ALL) ALL
```

A partir de maintenant, connectez-vous en tant que `user-ansible` sur le node manager.

Question 6. Vérification de la connexion SSH.
Depuis le node manager, connectez-vous en SSH au compte ***admin*** de chaque node. Ceci permet d'enregistrer sur le node manager l'empreinte SSH de chaque node, une étape indispensable pour pouvoir utiliser Ansible.

Question 7. Inventaire des nodes
La première chose à faire pour configurer Ansible est de réaliser l'**inventaire** des nodes. Un inventaire Ansible est un simple fichier texte au format INI. L'inventaire par défaut est le fichier `/etc/ansible/hosts`, mais vous pouvez utiliser n'importe quel fichier respectant la bonne syntaxe.

Le fichier `/etc/ansible/hosts` fourni lors de l'installation d'Ansible est un fichier d'exemple, vous pouvez vous en inspirer (et le modifier) pour créer votre inventaire.

Créez un dossier `ansible` dans `/home/user-ansible` sur le node manager, et créez votre fichier d'inventaire nommé `inventaire.ini` dans ce dossier. Le node `http1` sera placé dans un *groupe* nommé `apache` et le node `bdd1` sera placé dans un groupe nommé `db`.

Exercice 3. Première commande Ansible

Pour vous familiariser avec Ansible, vous allez commencer à l'utiliser en mode **ad-hoc** (c'est-à-dire avec des commandes plutôt qu'avec des scripts), pour mettre en place les prérequis.

Question 8. Commencez par lancer un *ping* sur les nodes à l'aide d'Ansible, avec la commande suivante :

```
$ ansible -i inventaire.ini -m ping all -u admin --ask-pass
```

Détaillons les options utilisées :

- `-i` : emplacement du fichier **i**nventaire ;
- `-m` : *module* à exécuter ;

- `-u` : compte utilisateur **présent sur la machine distante**
- `--ask-pass` : demande le mot de passe SSH (ici, le mot de passe de *admin* sur la machine distante).

⚠ Cette commande ne fonctionnera que si *admin* a un compte sur tous les nodes. Sinon, il faut remplacer l'option `all` par le nom des nodes.

Modules

Un **module** est un programme utilisé pour exécuter une tâche bien précise. Ainsi, chaque tâche utilise un module et un seul, qui peut prendre des arguments pour être exécuté de manière personnalisée. Ansible fournit de nombreux modules, disponibles sur la documentation Ansible ou à l'aide de la commande `ansible-doc --list` ; il est également possible de créer son propre module (dans n'importe quel langage).

Si tout va bien, vous devriez obtenir le résultat suivant :

```
http1 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}

bdd1 | SUCCESS => {
   "ansible_facts": {
       "discovered_interpreter_python": "/usr/bin/python3"
   },
   "changed": false,
   "ping": "pong"
}
```

💡 Le module `ping` ne demande aucune autorisation particulière, puisqu'il se contente de vérifier que les nodes sont bien joignables. Mais si l'on a besoin d'effectuer des opérations qui demandent des privilèges d'administrateur, on doit recourir à une *élévation de privilège*. Pour cela, on utilise l'option `--become` (ou simplement `-b`), combinée avec l'option `--ask-become-pass` pour saisir le mot de passe `sudo`.

Exercice 4. Création des comptes utilisateurs `user-ansible`

Dans l'exercice précédente, vous avez demandé à Ansible de s'identifier auprès des nodes à l'aide d'un compte utilisateur existant sur ceux-ci. Comme indiqué précédemment, il est préférable de disposer d'un utilisateur dédié à Ansible, de sorte que toutes les commandes Ansible seront loguées comme ayant exécutées par cet utilisateur. Mais à la différence du compte `user-ansible` créé sur le node manager, vous allez cette fois demander à Ansible de créer les utilisateurs sur les nodes pour vous !

Question 9. Commencez par générer la version chiffrée du mot de passe du compte :

```
$ ansible localhost -i inventaire.ini -m debug -a "msg={{ 'mot de passe' |
     password_hash('sha512', 'secretsalt') }}"
```

💡 L'option `localhost` indique à Ansible de lancer la commande en local, sur le node manager. Nous utilisons le module **debug** qui sert à afficher des messages d'informations. Ici, le message affiché est le résultat du chiffrement de la phrase *mot de passe* (à remplacer évidemment par le mot de passe de votre choix...). Le mot *secretsalt* est une donnée supplémentaire afin d'empêcher que deux informations identiques conduisent à la même *empreinte* (la résultante d'une fonction de hachage).

Question 10. Utilisez le mot de passé chiffré de la question précédente et le module *user* pour créer le nouvel utilisateur `user-ansible` sur les nodes (cherchez dans la documentation pour en savoir plus sur ce module !).

Les seuls arguments nécessaires du module **`user`** sont ici le nom et le mot de passe de notre nouvel utilisateur.

Si tout s'est bien passé, vous devriez obtenir un message de couleur orange, qui indique qu'une modification a eu lieu sur les nodes. Notez également la ligne `"password": "NOT_LOGGING_PASSWORD"` : le mot de passe de l'utilisateur ne figurera pas dans les logs sur le node.

Désormais, nous utiliserons les comptes `user-ansible` pour toutes les opérations sur les nodes. Il y a cependant un problème : sur les nodes, ces comptes n'ont pas encore les droits `sudo` !

Question 11. A l'aide du module *user*, *ajoutez* l'utilisateur `user-ansible` dans le groupe `sudo`.

On aurait bien entendu pu réaliser cette opération en même temps que la création du compte.

Vérifiez que l'utilisateur `user-ansible` a bien les droits `sudo` sur les nodes, en exécutant une commande `sudo`, par exemple :

```
$ ansible -i inventaire.ini all -m raw -a "sudo -l -U user-ansible" --user user-ansible
   --ask-pass --become --ask-become-pass
```

Pour l'instant, chaque commande Ansible exécutée sur un node demande de saisir le mot de passe SSH (et échoue si l'on ne spécifie pas l'option **`--ask-pass`**). C'est évidemment contraignant étant donné qu'on souhaite que les traitements soient réalisés de la manière la plus automatique possible. De plus, l'authentification par *clés SSH* est recommandée car elle est beaucoup plus sûre que l'utilisation d'un mot de passe.

Question 12. Sur le node manager, générez une paire de clés SSH de type ecdsa pour l'utilisateur `user-ansible` à l'aide de la commande `ssh-keygen -t ecdsa` (inutile de saisir une *passphrase*).

Clés SSH

Cette commande génère deux clés, une ***privée*** (conservée sur la machine de l'utilisateur) et une ***publique*** (déployée sur toutes les machines sur lesquelles il se connecte).

ECDSA (*Elliptic Curve Digital Signature Algorithm*) est un algorithme de signature numérique basé sur les *courbes elliptiuqes* qu'il est conseillé d'utiliser aujourd'hui pour avoir un niveau de sécurité satisfaisant.

La **passphrase** est utilisée pour renforcer la connexion SSH par un mot de passe. Dans le cadre du TP, pour gagner du temps nous ne l'utiliserons pas.

Il faut donc à présent déployer la clé publique sur les nodes. Vous allez pour cela utiliser le module Ansible **`authorized_key`**. Vous pourrez ainsi vous connecter sans saisir de mot de passe SSH.

Question 13. Saisissez la commande suivante pour déployer la clé publique :

```
$ ansible -i inventaire.ini all -m authorized_key -a 'user=user-ansible state=present
    key="{{ lookup("file", "/home/user-ansible/.ssh/id_ecdsa.pub") }}"'
    --user user-ansible --ask-pass
```

Le node manager et le node sont maintenant opérationnels ! Les exercices précédents vous ont permis d'aborder Ansible à l'aide des commandes ad-hoc, mais c'est dans les exercices suivants que vous allez pouvoir mesurer tout son potentiel, en réalisant des déploiements vraiment automatisés !

Exercice 5. Définitions des rôles

Rôles

Un **rôle** est une **structure arborescente** constituée de **répertoires** et de **fichiers de configuration** au format **YAML** (format que vous avez déjà vu avec **Netplan**), qui vont avoir pour fonction d'installer tel ou tel système (par exemple, le rôle d'installer Apache, le rôle d'installer MariaDB, etc.). Les rôles peuvent être imbriqués et interdépendants les uns des autres.

Tâches

Une **tâche** est une instruction décrite en YAML dans un fichier de configuration. Chaque tâche utilise un module ainsi que quelques éventuels arguments supplémentaires.

En résumé :

- un **rôle** contient un ou plusieurs **fichiers de configuration** (YAML) ;
- un **fichier de configuration** contient une ou plusieurs **tâches** ;
- une **tâche** fait appel à un **module**.

Vous allez donc transposer les étapes d'installation et de configuration décrites au début du sujet en rôles. Les cinq rôles à créer sont les suivants :

1. un rôle pour installer Apache : `apache` ;
2. un rôle pour installer MariaDB : `mariadb` ;
3. un rôle pour configurer Apache pour MediaWiki : `confapache`.
4. un rôle pour configurer MariaDB pour MediaWiki : `confdb`.
5. un rôle qui contiendra les variables globales : `commun`.

Question 14. Création du dossier qui contiendra les rôles
Créez dans votre dossier `ansible` un sous-dossier `roles`, et déplacez-vous à l'intérieur.

Il est possible d'indiquer un chemin vers les rôles qui ne se trouve pas dans le dossier courant, en modifiant la variable d'environnement `DEFAULT_ROLES_PATH` ou l'option `roles_path` du fichier `/etc/ansible/ansible.cfg`.

Question 15. Création du premier rôle
Comme indiqué ci-dessus, un rôle définit une arborescence de fichiers bien spécifique (exemple). Bien qu'il soit bien sûr possible de créer cette arborescence à la main, nous allons utiliser l'outil `ansible-galaxy` qui permet de télécharger, créer et gérer les rôles Ansible depuis le site web https://galaxy.ansible.com/.

Créez le rôle `apache` à l'aide de la commande :

```
$ ansible-galaxy init apache
```

Vous pouvez installer la commande `tree` afin de visualiser l'arborescence créée :

```
user-ansible@node-server:~/ansible/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
```

L'arborescence créée contient notamment les répertoires suivants :

- **`defaults`** : valeurs par défaut (le fichier **`main.yml`** est obligatoire) avec une priorité moindre ;
- **`files`** : tous les fichiers à copier sur le node ;
- **`handlers`** : liste les instructions handlers ; les handlers sont exécutés uniquement si une tâche les appelle à travers un **notify**, une notification déclenchée par la tâche (le fichier **`main.yml`** est obligatoire) ;
- **`meta`** : dépendances du rôle et informations (auteur, licence, plateformes...) sur le rôle (le fichier **`main.yml`** est obligatoire) ;
- **`tasks`** : liste des instructions à exécuter (le fichier **`main.yml`** est obligatoire) ;
- **`templates`** : tous les fichiers de template Jinja ;
- **`vars`** : fichier contenant des déclarations de variables (le fichier **`main.yml`** est obligatoire) ; les variables définies ici sont prioritaires par rapport aux variables définies dans l'inventaire.

En fait, pour installer Apache, la plupart de ces dossiers sont inutiles. **Supprimez de l'arborescence tous les dossiers sauf `handlers` et `tasks` (et leur contenu). Créez ensuite un fichier (vide) `php.yml` dans `tasks`.**

Question 16. Création du rôle `mariadb`
Inutile d'utiliser **`ansible-galaxy`** pour cela, **créez l'arborescence suivante :**

```
roles
└── mariadb
    └── tasks
        └── main.yml
```

Question 17. Création du rôle commun à Apache et MariaDB
Comme les opérations de configuration et les variables concernent spécifiquement le déploiement de MediaWiki, les rôles de configuration et commun vont être placés spécifiquement dans un répertoire **`mediawiki`**. Le rôle `commun` contiendra des variables partagées entre les rôles `confapache` et `confdb` (définis ci-dessous). Ainsi, au lieu de définir les mêmes variables à deux endroits différents, il est préférable de créer un rôle commun, et d'utiliser une dépendance avec les autres rôles.

Créez le rôle `commun` défini comme suit :

```
roles
└── mediawiki/
    └── commun
        └── defaults
            └── main.yml
```

Question 18. Création des rôles `confdb` et `confapache`
Pour le rôle `confdb`, vous avez besoin de **créer une base de données** et d'**attribuer des droits** sur cette base. Et vous créerez une dépendance avec le rôle **`commun`** pour partager des **variables globales**. Pour cela, vous avez besoin des répertoires `meta` et `tasks`, et des fichiers main.yml qui contiendront les actions.

Créez le rôle `confdb` défini comme suit :

```
mediawiki
└── confdb
    ├── meta
    │   └── main.yml
    └── tasks
        └── main.yml
```

De manière similaire, créez le rôle `confapache` comme suit :

```
mediawiki
└── confapache
    ├── meta
    │   └── main.yml
    └── tasks
        └── main.yml
```

Vous devriez donc disposer à ce stade de l'arborescence suivante :

```
roles
├── apache
│   ├── handlers
│   │   └── main.yml
│   └── tasks
│       ├── main.yml
│       └── php.yml
├── mariadb
│   └── tasks
│       └── main.yml
└── mediawiki
    ├── commun
    │   └── defaults
    │       └── main.yml
    ├── confapache
    │   ├── meta
    │   │   └── main.yml
    │   └── tasks
    │       └── main.yml
    └── confdb
        ├── meta
        │   └── main.yml
        └── tasks
            └── main.yml
```

Vous allez à présent pouvoir créer les tâches, en complétant les différents fichiers `.yml`.

Exercice 6. Création des tâches

Un fichier `.yml` commence toujours par **trois tirets** (`---`). Les différentes tâches commencent ensuite par **un 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 **deux espaces**.

Question 19. Création des tâches d'installation d'Apache

Créez le fichier `roles/apache/tasks/main.yml` suivant :

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

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

#3. Cette tâche fait appel à un autre fichier de configuration pour installer PHP.
#   Elle est exécutée uniquement si la variable php_install est à vraie (par défaut,
#   elle est à faux)
- name: "install php8.0 packages"
  include_tasks: "php.yml"
  when: php_install|default(False)|bool
```

Quelques explications :

- la première tâche fait appel au module Ansible **`apt`** pour gérer le package **`apache2`** ; l'option **`state: present`** indique qu'il faut l'installer. Le nom de la tâche (**`apache installation`**) apparaîtra dans les logs générés par Ansible.
- la deuxième tâche utilise le module Ansible **`systemd`** pour démarrer le service **`apache2`** ; celui-ci sera également démarré à chaque démarrage de la machine, grâce à l'option **`enabled: yes`**. Cette tâche doit donc être effectuée après la première.
- la troisième tâche inclut un fichier externe pour l'installation des paquets PHP, ce qui contribue à renforcer la modularité de notre installation. La condition **`when`** avec le **filtre** (**`php_install|default(False)|bool`**) permettent de conditionner l'installation de PHP. La syntaxe utilisée signifie que la variable **`php_install`** est un booléen par défaut à **`false`**. Cette variable sera initialisée plus tard.

Question 20. Création de la tâche d'installation de PHP

Créez le fichier `roles/apache/tasks/php.yml` suivant :

```
---

#1. Cette tâche installe PHP8 et ses extensions
- name: "install php8.0 packages"
  apt:
    name: "php, php-mysql, php-xml, php-mbstring, php-gd, php-intl"
    state: latest
  changed_when: yes
  notify: [ "apache restart" ]
```

- **`state: latest`** indique d'installer les dernières versions des paquets disponibles
- **`changed_when`** force à signaler un changement d'état ;
- **`notify: [ "apache restart" ]`** indique que si la tâche change d'état (et uniquement si la tâche a engendré un changement), **`notify`** fait appel au *handler* "apache restart" pour relancer le service Apache.

Question 21. Création du handler de redémarrage d'Apache

Créez le fichier `roles/apache/handlers/main.yml` suivant :

```
---
# handlers file for apache

- name: "apache restart"
  systemd:
    name: "apache2"
    state: "restarted"
```

Question 22. Création des tâches d'installation de MariaDB

Créez le fichier `roles/mariadb/tasks/main.yml` suivant :

```
---

# MariaDB packages installation
- name: "mariadb-server installation"
  apt:
    name:  "mariadb-server, python3-mysqldb"
    state: "present"

# Enable connections to MariaDB from outside
- name: "mariadb authorized hosts"
  ini_file:
    dest: /etc/mysql/mariadb.conf.d/99-custom.cnf
    section: mysqld
    option: "{{ item.option }}"
    value: "{{ item.value }}"
  with_items:
    - { option: bind-address, value: "0.0.0.0" }

# MariaDB service starting
- name: "start mariadb service"
  systemd:
    name:  "mariadb"
    state: "started"
    enabled: yes
```

La deuxième tâche fait appel au module **`ini_file`** pour créer un fichier de configuration au format **`INI`** (format utilisé par MariaDB). En effet, par défaut, sous Debian et Ubuntu, MariaDB interdit les connexions aux bases de données depuis une autre machine, ce qui est concrétisé par la ligne **`bind-address = 127.0.0.1`** du fichier **`/etc/mysql/mariadb.conf.d/50-server.cnf`**. La documentation indique que les différents fichiers de ce dossier sont lus dans l'ordre, et que c'est la dernière valeur d'une variable qui sera prise en compte. On crée donc un fichier nommé **`99-custom.cnf`** avec la valeur **`bind-address = 0.0.0.0`**, qui indique que les connexions sont autorisées depuis n'importe quelle machine (probablement pas idéal en terme de sécurité, mais cela sort du cadre de ce TP).

Question 23. Création des tâches pour le rôle `commun`

Le rôle **`commun`** crée des variables globales qui seront utilisées dans les rôles **`confapache`** et **`confdb`**. Pour connaître les variables à définir, il faut étudier le guide d'installation et repérer ce qui est potentiellement une variable. Très souvent, il s'agit de noms de répertoires, de bases de données ou des logins, mots de passe, etc.

Au format YAML, les variables sont définies par un couple "**clé : valeur**" séparé par deux points (**`:`**). Pour exploiter ensuite une variable YAML, il faut la placer entre accolades.

La première chose à faire est de **chiffrer les mots de passe**, afin que ceux-ci n'apparaissent pas en clair dans les fichiers de tâche. Pour cela, vous allez utiliser la commande **`ansible-vault`**, qui permet de conserver les mots de passe dans un "coffre-fort", lui même protégé par une clé de chiffrement. Pour ce TP, vous utiliserez la clé de chiffrement **`Ansible`**.

Il faut ensuite définir deux mots de passe : le **mot de passe de la base de données**, et le **mot de passe d'administration de MediaWiki**. Pour ce TP, les mots de passe seront identiques, et définis à **`cpelyon2022`**.

MediaWiki impose un mot de passe d'**au moins 10 caractères**.

Saisissez la commande suivante :

```
ansible-vault encrypt_string 'cpelyon2022' --name 'mediawiki_db_password'
```

L'option **`--name`** permet d'obtenir le nom de la variable spécifiée (ici **`mediawiki_db_password`**) dans la sortie.

Tâches

Il est évidemment extrêmement dangereux de renseigner les mots de passe en clair, directement sur la ligne de commande comme nous venons de le faire, puisque les commandes sont historisées. La commande **`ansible-vault`** dispose heureusement de méthodes plus sécurisées, comme la spécification de fichiers de paramètres.

Si tout se passe bien, vous devez obtenir un résultat semblable à celui-ci :

```
mediawiki_db_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          3636393130613832656562656339343865313135666633386461343062366236326630623136303 3
          6366366266646332633133656535666534343866383731630a343530626632373866626165326437
          6137373461316236633238323363383736376439333834356533353635613436313863396563623 7
          3766393133303534610a3065313630353965616635323762636539333732353839643566366437 36
          6434
Encryption successful
```

Recommencez afin de définir la variable `mediawiki_admin_password`.

Copiez les sorties de **`ansible-vault`** dans un fichier texte temporaire, vous en aurez besoin dans quelques secondes.

Créez à présent le fichier `roles/mediawiki/commun/defaults/main.yml` suivant :

```
---
# nom de la base de données
mediawiki_db_name: "mediawiki"

# nom de l'utilisateur de la base de données et son mot de passe
mediawiki_db_user: "mediawiki"
mediawiki_db_password: !vault |
```

```
          $ANSIBLE_VAULT;1.1;AES256
          38376164663038393134643734663163343362616263346563623837313532643337636163323133
          6131653863346432666132353434383330353762396262310a343833336462346165386431323965
          32613265353764336462646638346462303761303766383238323931356637663563306466353031
          3331343966303565340a373633616233643966633433323535636332643338663237613466613935
          6230

# nom et mot de passe de ladministrateur Mediawiki
mediawiki_admin_user: "admin"
mediawiki_admin_password: !vault |
          $ANSIBLE_VAULT;1.1;AES256
          36353734646533626431616238363135636163666233323366313761393465373032346535656330
          3331336439366365343031323536363430613935653265650a326437613766633664313536386637
          30303933373962356566616534333263343730616239643063313265626366376235343962663236
          6435316436333638650a626363346363383266386564356437356133383330636661333630643166
          3033

# nom du Mediawiki et son titre
mediawiki_name: "mediawiki"
# Les apostrophes sont nécessaires en raison des espaces
mediawiki_title: "'CPE Lyon Wiki'"

# emplacement du répertoire d'installation de Mediawiki
mediawiki_directory: "/var/www/html/{{mediawiki_name}}"

# répertoire de maintenance de Mediawiki
mediawiki_maintenance_directory: "{{mediawiki_directory}}/maintenance"

# premier node du groupe mariadb
mediawiki_db_host: "{{groups.db.0}}"

# url des sources Mediawiki
mediawiki_archive_url: "https://releases.wikimedia.org/mediawiki/1.37/mediawiki-1.37.1.tar.gz"
```

⚠ Les mots de passe chiffrés sont bien sûr à remplacer ici par ceux que vous avez obtenus !

💡 MediaWiki sera installé à partir des sources. On aurait évidemment pu là aussi utiliser le paquet de notre distribution, mais l'idée était de découvrir une autre méthode. On a également ainsi l'avantage de disposer de la *dernière* version de MediaWiki.

Question 24. Création des tâches pour le rôle `confdb`

Pour que le rôle **`confdb`** puisse utiliser les variables enregistrées dans le rôle **`commun`**, il faut déclarer cette dépendance dans le répertoire **`meta`** :

Complétez le fichier `roles/mediawiki/confdb/meta/main.yml` comme suit :

```
dependencies:
  - role: "mediawiki/commun"
```

Créons à présent la base de données et un utilisateur de cette base.

Complétez le fichier `roles/mediawiki/confdb/tasks/main.yml` comme suit :

```
---

#1. Installation de la base de données Mediawiki
```

```
- name: "mediawiki database"
  mysql_db:
    name: "{{mediawiki_db_name}}"
    state: present

#2. Création dun accès utilisateur et attribution des privilèges sur la base Mediawiki
- name: "mediawiki user+privileges"
  community.mysql.mysql_user:
    name: "{{mediawiki_db_user}}"
    password: "{{mediawiki_db_password}}"
    priv: "{{mediawiki_db_name}}.*:ALL,GRANT"
    host: "{{item}}"
    state: present
  with_items: "{{groups.apache}}"
```

Question 25. Création des tâches pour le rôle `confapache`

Nous touchons bientôt au but ! Nous allons tout d'abord déclarer la dépendance au rôle **`commun`** comme précédemment.

Complétez le fichier `roles/mediawiki/confapache/meta/main.yml` comme suit :

```
dependencies:
  - role: "mediawiki/commun"
```

Il nous reste à créer les tâches de configuration d'Apache pour MediaWiki.

Complétez le fichier `roles/mediawiki/confapache/tasks/main.yml` comme suit :

```
---
#1. Le package acl est requis pour éviter une erreur de droits dans la tâche 4
- name: "Install ACL package"
  apt:
    name: "acl"
    state: present

#2. Création du repertoire pour l'installation des fichiers Mediawiki
- name: "mediawiki directory"
  file:
    path: "{{mediawiki_directory}}"
    owner: "www-data"
    group: "www-data"
    state: directory

#3. Décompresse le fichier source archive Mediawiki et le formate sans extension
- name: "uncompress mediawiki archive"
  unarchive:
    src: "{{mediawiki_archive_url}}"
    dest: "{{mediawiki_directory}}"
    owner: "www-data"
    group: "www-data"
    remote_src: yes
    # supprime mediawiki-1.xx.x/ du chemin
    extra_opts: --transform=s/mediawiki-[0-9\.]*\///

#4. Exécute la tâche avec l'utilisateur user-ansible, se place dans le répertoire
    de maintenance et exécute la commande de configuration si le fichier
```

```
    LocalSettings.php nexiste pas
- name: "mediawiki configuration"
  become: yes
  become_user: "user-ansible"
  args:
    creates: "{{mediawiki_directory}}/LocalSettings.php"
    chdir: "{{mediawiki_maintenance_directory}}"
  command:
    php install.php --scriptpath /{{mediawiki_name}}
    --dbname mediawiki --lang fr
    --dbuser {{mediawiki_db_user}}
    --dbpass {{mediawiki_db_password}}
    --pass {{mediawiki_admin_password}}
    --dbserver {{mediawiki_db_host}}
    {{mediawiki_title}} {{mediawiki_admin_user}}
  run_once: yes
  delegate_to: "{{item}}"
  with_items: "{{groups.apache}}"

#5. Exécute la tâche avec l'utilisateur user-ansible, se place dans le répertoire de
    maintenance et exécute la commande de mise à jour de la base une seule fois
- name: "mediawiki db update"
  become: yes
  become_user: "user-ansible"
  command:
    php update.php --quick
  args:
    chdir: "{{mediawiki_maintenance_directory}}"
    # La mise à jour à besoin d'être lancée une seule fois
  run_once: yes
  register: resultat
  changed_when: "' ...done.' in resultat.stdout"
```

Exercice 7. Etape finale : définition des playbooks pour automatiser le déploiement

Ce dernier exercice a pour but d'assembler toutes les opérations effectuées précédemment et d'automatiser le déploiement de MediaWiki. Pour cela, vous allez utiliser les **playbooks** Ansible.

Playbooks

Un playbook est un **fichier de configuration YAML** contenant une suite de jeux d'instructions (*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.

Vous allez donc créer 3 playbooks :

- un pour installer Apache,
- un pour installer MariaDB,
- un pour configurer MediaWiki.

⚠ L'ordre d'exécution de ces playbooks est important, car les rôles de configuration dépendent des rôles d'installation. Par exemple, les serveurs Apache et MariaDB doivent être installés **avant** que la configuration de MediaWiki puisse être lancée.

Question 26. Création du playbook pour installer Apache

Créez le playbook **install-apache.yml** à la racine du dossier **ansible** et saisissez son contenu :

```
---
- name: "Installation apache"
  hosts: http1
  roles:
    - role: "apache"
      php_install: yes
```

On retrouve les instructions suivantes :

- **hosts: http1** indique le node concerné ;
- **role: "apache"** indique le rôle à lancer ;
- **php_install: yes** indique la valeur de la variable **php_install**.

Il est temps de lancer le playbook ! Saisissez la commande suivante :

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

Vous devriez voir sur la sortie une tâche **[Gathering Facts]**. Il s'agit d'une tâche par défaut au lancement d'un playbook, chargée de récupérer des informations (adresse IP, version de l'OS, etc.) et les variables d'environnement de chaque node (des **facts**). Il est possible de désactiver cette tâche (voir question suivante).

Si tout s'est bien passé, PHP et le serveur Apache devraient être installés sur le node **http1**. Pour le vérifier, saisissez les commandes suivantes sur **http1** :

```
$ php --version
$ systemctl status apache2
```

Pour tester votre serveur Apache, ajoutez une redirection du port 80 de votre machine hôte (machine physique) vers le port 80 du serveur **http1**, puis saisissez l'url **localhost** dans un navigateur sur la machine hôte. Vous devriez voir la page par défaut d'Apache 2.

Question 27. Création du playbook pour installer MariaDB

L'installation de MariaDB consiste simplement à lancer le rôle **mariadb**. Créez le playbook **install-mariadb.yml** à la racine du dossier **ansible** et saisissez son contenu :

```
---
- name: "Installation MariaDB"
  hosts: bdd1
  gather_facts: no
  roles:
    - role: mariadb
```

Remarquez qu'ici nous avons désactivé la tâche **gather_facts** (pas utile dans notre cas).

Lorsqu'un playbook est volumineux ou que les tâches à exécuter sont longues, il peut être utile de n'exécuter que certaines parties de ce playbook, identifiées par un ***tag***.

Exécutez ce playbook avec la commande suivante :

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

Vérifiez que MariaDB est effectivement et correctement installé, en saisissant sur **bdd1** :

```
$ systemctl status mariadb
```

Question 28. Création du playbook pour installer MediaWiki

Enfin, l'installation et la configuration de MediaWiki consistent à lancer les rôles **confapache** et **confdb**. Créez le playbook **install-mediawiki.yml** à la racine du dossier **ansible** et saisissez son contenu :

```
- name: "MediaWiki db configuration"
  hosts: db
  gather_facts: no
  tags: [ "mariadb", "mysql" ]
  roles:
    - role: "mediawiki/confdb"

- name: "MediaWiki apache configuration"
  hosts: apache
  gather_facts: no
  tags: "apache"
  roles:
    - role: "mediawiki/confapache"
```

Exécutez ce playbook avec la commande suivante :

```
$ ansible-playbook -i inventaire.ini --user user-ansible --become --ask-become-pass
    --ask-vault-pass install-mediawiki.yml
```

On remarque ici l'ajout de l'option `--ask-vault-pass` afin de pouvoir déchiffrer les mots de passe stockés dans le coffre-fort.

Si vous rencontrez une erreur de caractère non hexadécimal, supprimez les éventuels espaces présents à la fin de chaque ligne du résultat de `ansible-vault` dans le fichier **roles/mediawiki/commun/default/main.yml**.

Voilà ! Au terme d'un TP long et éreintant, vous pouvez taper dans un navigateur sur votre machine hôte l'adresse **localhost/mediawiki** et admirer votre propre installation de MediaWiki, que vous allez pouvoir répliquer et adapter à volonté ! Vous pouvez également essayer de vous connecter en admin sur MediaWiki.