Rapport Technique

Projet Ansible LAMP

Pour François Delrue

Auteur : Abdel - Vincent - Rodolphe

Table des matières

Introduction

Consigne

Travail en groupe de 3 maximum.

A partir de votre node manager Ansible, vous devez créer un playbook/rôle permettant de :

- Déployer une stack LAMP **containeurisée** sous Docker
  - Pouvoir contrôler la stack via Ansible
    - Build les Dockerfile nécessaire au déploiement de la stack
    - Modifier les containers depuis le node manager via les Dockerfile
    - Choisir les ressources statiques (Site web)
    - Start/Stop/Restart
  - Web :
    - Sur un serveur dédié à la partie Web
    - Container Apache/Nginx
    - Container PHP séparé pour servir tous les containers Web
  - Database :
    - Container Mysql sur serveur séparé de la partie web
    - Avoir de la *Persistence des données*

- Le rôle Ansible devra pouvoir déployer une Stack LAMP sur des machines fraichement créées sans aucune configuration préalable autre que la connexion SSH et l'installation d'un interpréteur python.

Stack LAMP Containerisée

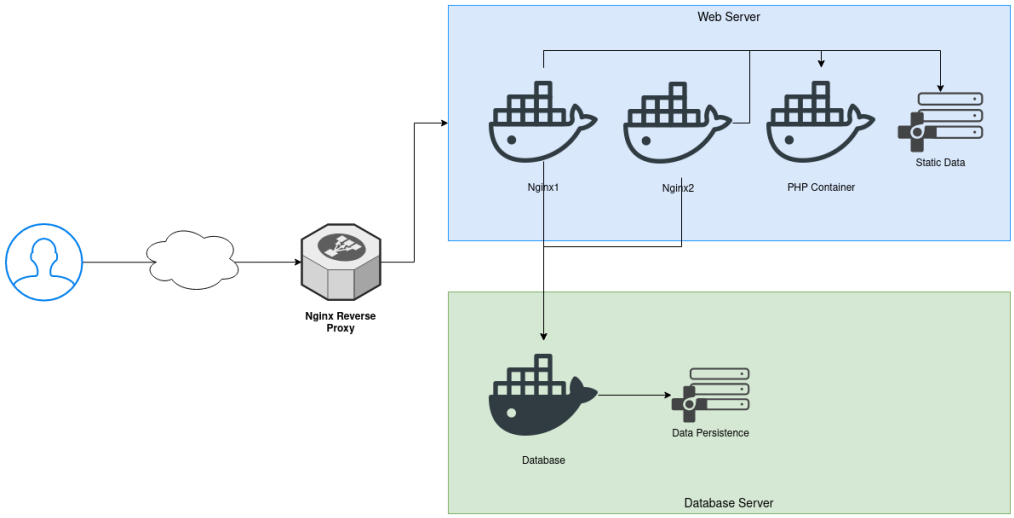
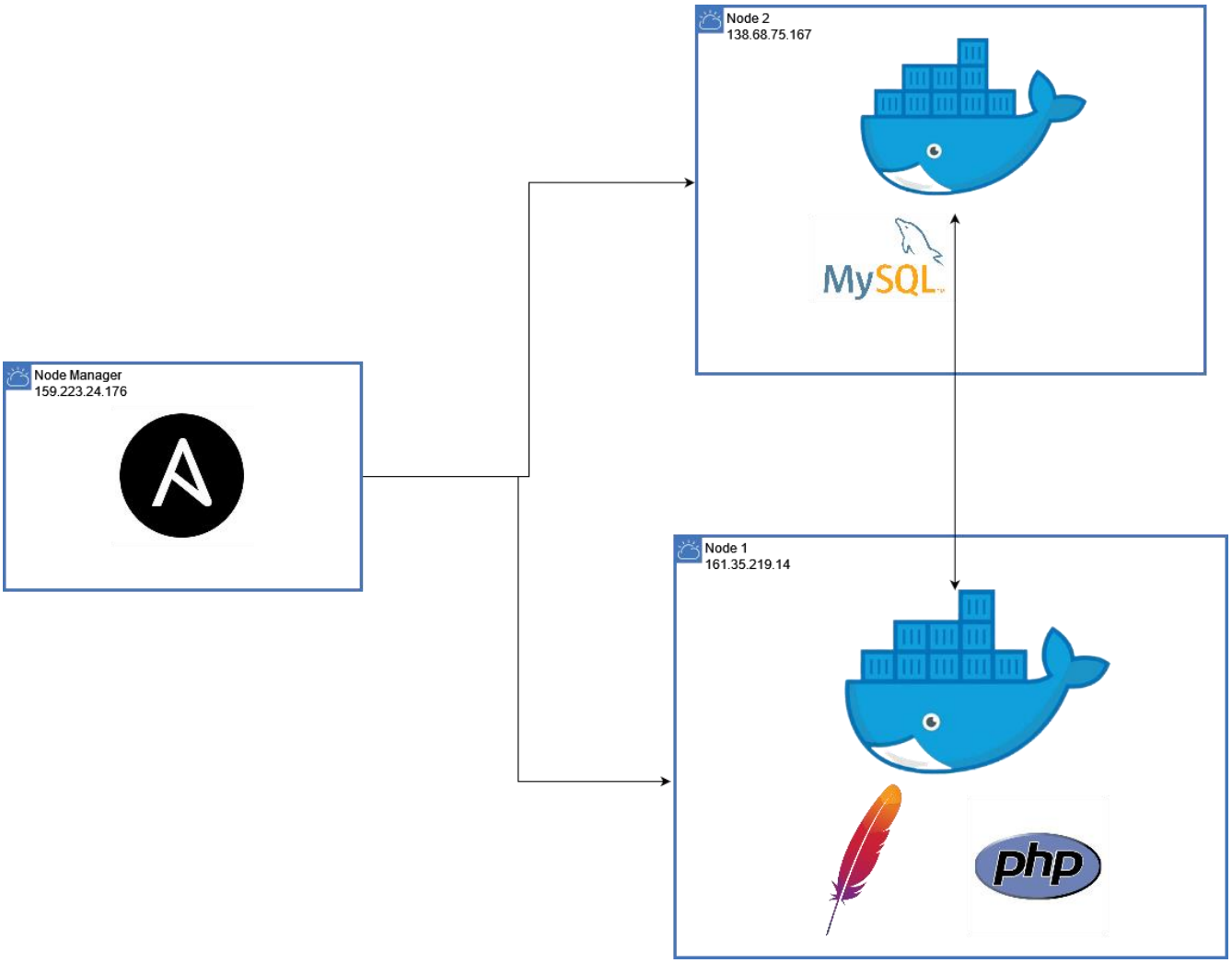


Schéma d'infrastructure



Résumé de l'automatisation

Le but de ce projet est de déployer une infrastructure automatisée.

Nous avons donc travaillé sur un cluster de Virtual Machine héberger sous Ubuntu 20.04.

Pour ce qui est de l'orchestration on aura utilisé Ansible qui nous permettra d'installer un Serveur Web Apache lié à PHP de manière containeurisé sous Docker.

De ce fait on aura aussi mis en place un Serveur BDD sous MySQL containeurisé sous Docker également pour permettre l'échange et le stockage des données.

Pour ce qui est de l'infrastructure on aura donc 3 Server Ubuntu.

Serveur :

- Manager : 159.223.24.176
- Node 1 : 161.35.219.14
- Node 2 : 138.68.75.167

Parmi les 3 machines une sera considérer comme notre Server Manager (Master). Les deux autres seront des Nodes (Slaves) ou y seront déployer les différents services.

Bien entendu en termes de prérequis on aura installé OpenSSH sur chacune des machines.

Autoriser la connexion SSH en autorisant l'authentification par le moyen de PublicKey et l'utilisateur Root sur notre Server Manager.

Ensuite on aura dû générer une clé SSH sur notre Server Manager qu'on aura déployé sur les différentes Nodes.

Installation :

Au niveau de notre Server Manager on aura donc installé Ansible avec la commande ci-dessous.

```
sudo python3 -m pip install ansible
```

Sur les différentes Nodes sera installer PIP voir la commande ci-dessous.

```
sudo apt-get install python3-pip python-dev
```

Création du fichier host

On aura donc créé un fichier de configuration nommé « hosts.yml » à la racine de notre dossier projet.

Le but de ce fichier est tout simplement de déclarer les différentes IP et/ou les noms d'hôtes de nos différentes nodes.

Ci-dessous vous pourrez retrouver le contenu du fichier hosts.yml :

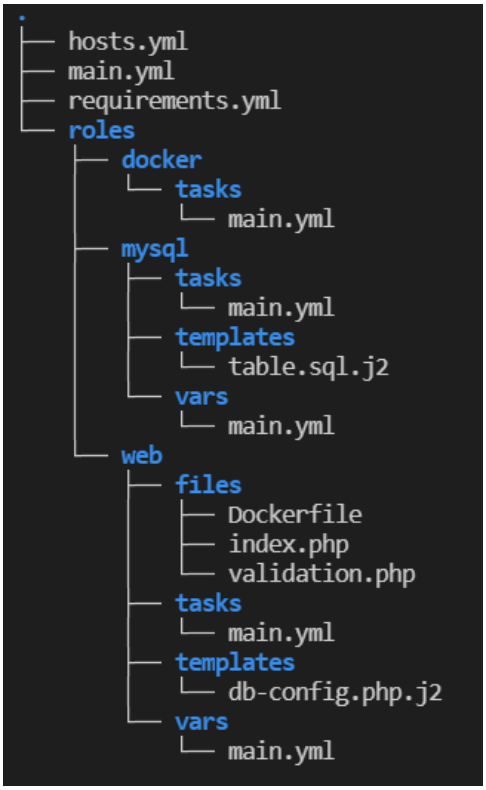
```
[web]
node.1

[db]
node.2
```

Création de l'arborescence des dossiers

Au niveau de l'arborescence on pourra donc y retrouver un dossier « roles » contenant 3 sous dossiers.

Un dossier docker, mysql et web.

Chacun des dossiers contient des playbooks et/ou dockerfile et également des fichiers pour les requis.



Partie Web

- Côté serveur web :
  - Build d'une image nginx/apache personnalisée
  - Déploiement d'une image php
  - Déployer les sources de notre application dans notre serveur web distant
  - S'assurer que le service Web est bien démarré
  - Docker-compose peut être utilisé pour la stack Web.

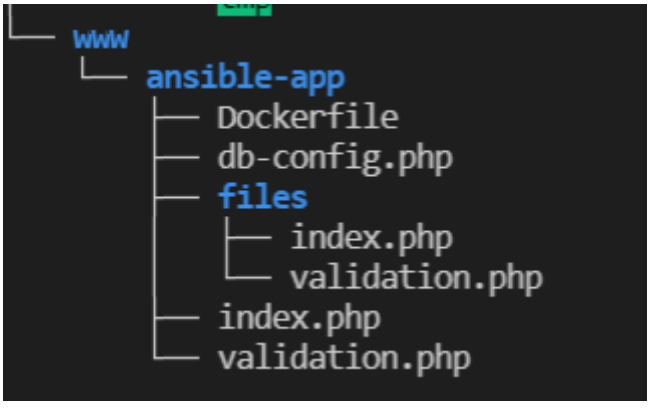
Fichier variable pour la partie Web :

```
mysql_user: "admin"
mysql_password: "secret"
mysql_dbname: "blog"
db_host: XXX.XXX.XXX.XXX
webserver_host: XXX.XXX.XXX.XXX
```

Les variables sont principalement utilisées dans les fichiers templates fournis.

- mysql_user : l'utilisateur de notre base de données mysql qui exécutera nos requêtes SQL depuis notre application web.
- mysql_password : le mot de passe de l'utilisateur de notre base de données mysql.
- mysql_dbname : le nom de notre base de données.
- db_host : l'ip de notre machine mysql (utile pour la partie configuration mysql de notre application web).
- webserver_host : l'ip de la machine web (utile pour autoriser uniquement l'ip du serveur web à communiquer avec notre base de données).

Voici l'arborescence que l'on retrouve sur la Node 1 :



Dockerfile

Vous pourrez retrouver le Dockerfile que l'on a élaboré pour la partie Web ci-dessous.

A savoir on a donc rentré la commande pour l'update des paquets avec « apt update » suivis de l'installation de apache 2 & php extension php-mysql avec la commande « apt install ».

On aura aussi effectué une attribution de droit 777 sur /var/www/html grâce à la commande « chmod » et une suppression du fichier index.html contenu dans /var/www/html avec la commande « rm ».

```
FROM ubuntu:18.04

ENV DEBIAN_FRONTEND noninteractive


RUN apt update && \
    apt install apache2 php php-mysql -y && \
    chmod 777 /var/www/html && \
    rm /var/www/html/index.html

EXPOSE 80
EXPOSE 443

CMD apachectl -D FOREGROUND
```

Playbook

Ci-dessous vous pourrez retrouver le playbook.yml utilisé pour le déploiement de la partie Web.

```yaml
---

- name: Create App directory
  ansible.builtin.file:
    state: directory
    path: /var/www/ansible-app
    mode: 0755

- name: Copy template file
  ansible.builtin.template:
    src: templates/db-config.php.j2
    dest: /var/www/ansible-app/db-config.php

- name: Copy app files
  ansible.builtin.copy:
    src: files/
    dest: /var/www/ansible-app/


- name: Create custom image for apache and php
  community.docker.docker_image:
    name: my-apache
    build:
      path: /var/www/ansible-app
    source: build

- name: Create container apache
  community.docker.docker_container:
    name: web
    image: my-apache
    state: started
    restart_policy: on-failure
    ports:
      - "80:80"
    volumes:
    - /var/www/ansible-app:/var/www/html
```

Explication étape par étape

Le name : Correspond à quelle étape en est le playbook c'est une manière de séquencer chaque tâche. Il s'affiche de manière visuelle dans le terminal.

Tâche 1 : permet de créer un répertoire ansible-app sur le serveur Web.

Tâche 2 : consiste à venir faire une copie du fichier template nommé db-config.php contenue dans un chemin déclaré dans SRC (source) pour venir le copier dans le DST (destination).

Tâche 3 : permet de faire la copie du fichier dans un dossier déclaré.

Tâche 4 : permet de crée une image custom pour apache et php (ici notre image my-apache)

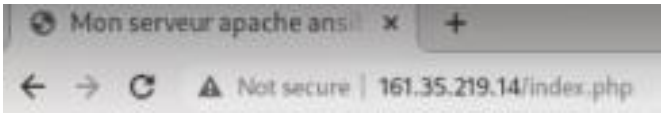
Tâche 5 : elle crée le container apache en utilisant l'image my-apache, restart le container en cas de problème. Elle utilisera le port 80 :80 et sont volumes sera déclaré dans le chemin prédéfini.

Déploiement Stack Web

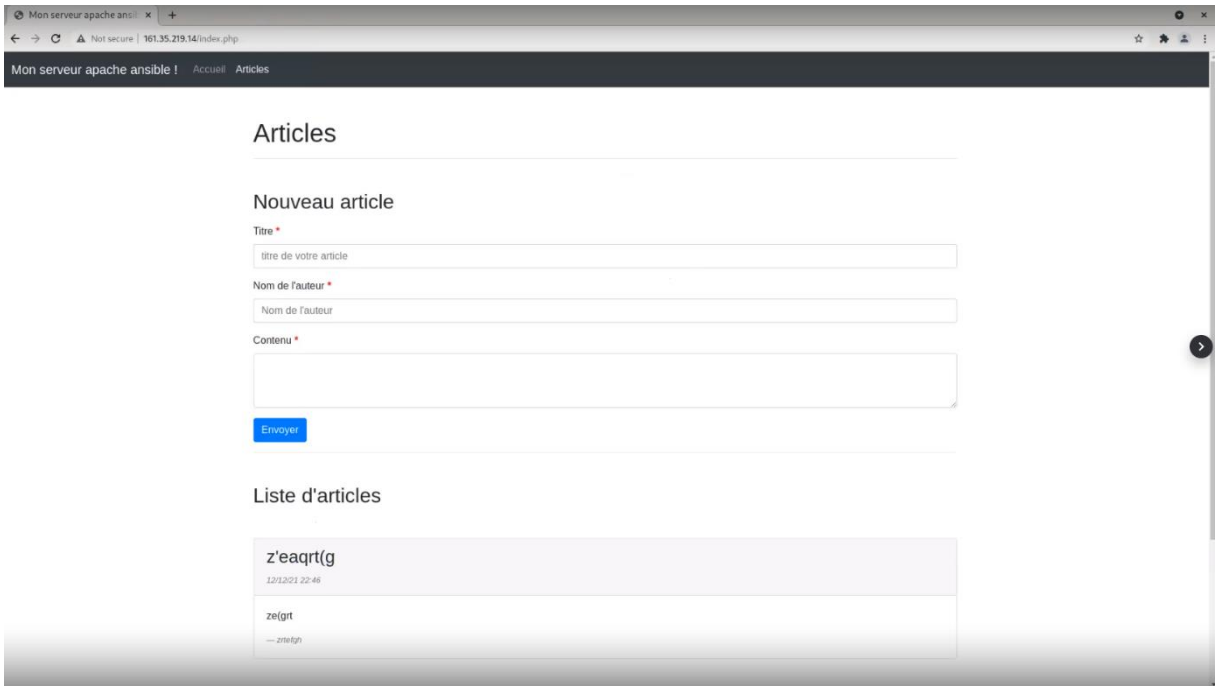
```
PLAY [web] *********************************************************

TASK [Gathering Facts] *********************************************
ok: [node.1]

TASK [web : Create App directory] **********************************
ok: [node.1]

TASK [web : Copy template file] ************************************
ok: [node.1]

TASK [web : Copy app files] ****************************************
ok: [node.1]

TASK [web : Create custom image for apache and php] ***********
ok: [node.1]

TASK [web : Create container apache] *******************************
ok: [node.1]
```

Résultats

Une fois la stack déployer nous allons nous rendre sur notre navigateur et rentrer l'adresse IP de notre serveur Web voir l'image ci-dessous.



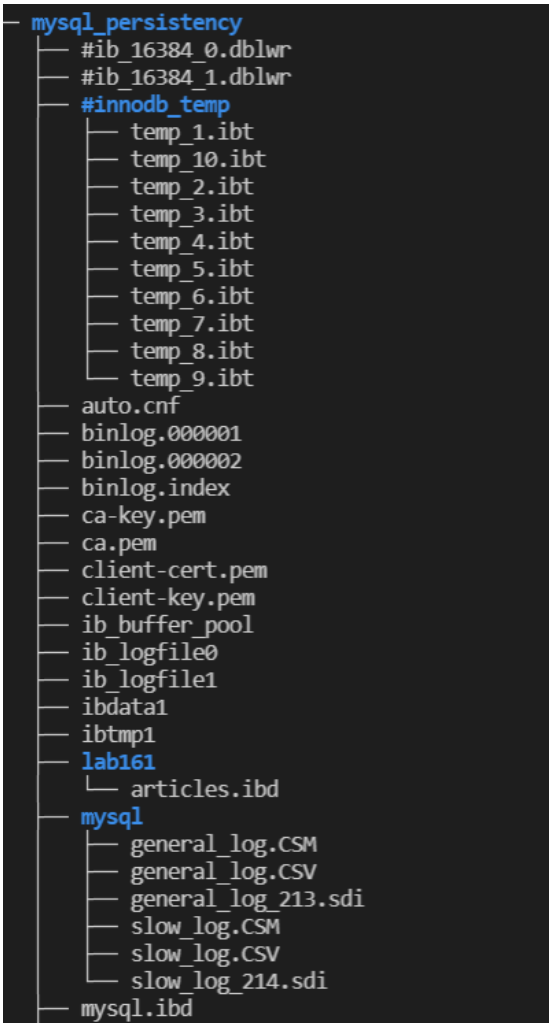
On pourra donc constater qu'il arrive bien à le joindre.



Partie Database

- Côté serveur base de données :
  - Installer les packages mysql et les paquets nécessaire au fonctionnement du module mysql d'Ansible (Voir la doc).
  - Modifier le mot de passe root
  - Autoriser notre serveur web à communiquer avec la base de données
    - Dans la configuration MySQL
      : `/etc/mysql/mysql.conf.d/mysqld.cnf` commenter les lignes :
      - `bind-address`
      - `skip-external-locking'`
  - Configurer notre table mysql avec les bonnes colonnes et autorisations

Voici l'arborescence que l'on retrouve dans la Node 2 :



Playbook

```yaml
---

- name: Pull an image mysql
  community.docker.docker_image:
    name: mysql
    source: pull

- name: Create Mysql persistence directory
  ansible.builtin.file:
    state: directory
    path: /var/mysql_persistency
    mode: 0755

- name: Create container mysql
  community.docker.docker_container:
    name: main_db
    image: mysql
    state: started
    restart_policy: on-failure
    ports:
      - "3306:3306"
    env:
      MYSQL_ROOT_PASSWORD: "{{ mysql_root_password }}"
      MYSQL_USER: "{{ mysql_user }}"
      MYSQL_PASSWORD: "{{ mysql_password }}"
      MYSQL_ROOT_HOST: "%"
    volumes:
      - /var/mysql_persistency:/var/lib/mysql

- name: Install PyMySQL python library
  pip:
    name: PyMySQL

- name: Install mysql bin client and mysqldump
  apt:
    name:
      - mysql-client
    update_cache: yes
```

```
- name: Grant user right
  community.mysql.mysql_user:
    name: "{{ mysql_user }}"
    priv: "*.*:ALL"
    login_user:  root
    login_password: "{{ mysql_root_password }}"
    login_port: 3306
    login_host: 10.114.0.7

- name: Query alter user
  community.mysql.mysql_query:
    login_user: root
    login_password: "{{ mysql_root_password }}"
    login_port: 3306
    login_host: 127.0.0.1
    query:
      - ALTER USER '{{ mysql_user }}'@'%' IDENTIFIED WITH
mysql_native_password BY '{{ mysql_password }}';
      - GRANT ALL ON *.* TO 'admin'@'%'
      - FLUSH PRIVILEGES;

- name: Copy database sql file
  ansible.builtin.template:
    src: templates/table.sql.j2
    dest: /tmp/table.sql

- name: Create database with set table
  community.mysql.mysql_db:
    login_user: root
    login_password: "{{ mysql_root_password }}"
    login_port: 3306
    login_host: 127.0.0.1
    name: "{{ mysql_database }}"
    state: import
    target: /tmp/table.sql
```

Explication étape par étape

Le name : Correspond à quelle étape en est le playbook c'est une manière de séquencer chaque tâche. Il s'affiche de manière visuelle dans le terminal.

Tâche 1 : on récupère l'image de mysql pour docker sur le hub

Tâche 2 : on crée un répertoire persistant pour mysql

Tâche 3 : création du container docker mysql et on y indique les paramètres comme le volume pour la persistance des données.

Tâche 4 : installation de librairie python pour mysql

Tâche 5 : installation de mysql bin client et de mysqldump (sauvegarde)

Tâche 6 : on y accorde des droits utilisateurs

Tâche 7 : on passe une requête pour modifier l'utilisateur

Tâche 8 : on effectue une copie du fichier sql de la database d'un répertoire source vers un répertoire destination.

Tâche 9 : création de la base de données avec des tables que l'on importe depuis un fichier sql.

Déploiement stack BDD

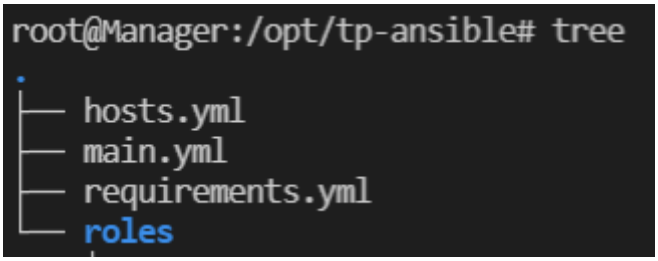
```
PLAY [db] **********************************************************************

TASK [Gathering Facts] *********************************************************
ok: [node.2]

TASK [mysql : Pull an image mysql] *********************************************
ok: [node.2]

TASK [mysql : Create container mysql] ******************************************
ok: [node.2]

TASK [mysql : Install PyMySQL python library] **********************************
ok: [node.2]

TASK [mysql : Install mysql bin client and mysqldump] **************************
ok: [node.2]

TASK [mysql : Grant user right] ************************************************
changed: [node.2]
```

Playbook

Le Playbook global

Pour bien ordonnancer tous les playbooks vu précédemment dans un certain ordre et les exécuter les uns après les autres on aura donc créé un playbook nommé main.yml global à la racine de notre arborescence qui vient appeler chacun des rôles (web & et bdd).

Voici un screen montrant ou se situe le main.yml dans l'arborescence :



Voici le code contenu dans le playbook :

```yaml
---
- hosts: all
  roles:
  - docker

- hosts: db
  roles:
  - mysql

- hosts: web
  roles:
  - web
```

Installation de Docker

Un des prérequis qu'on a eu besoin d'installer c'est tout simplement Docker. On aura créé un playbook nommé main.yml pour effectuer une installation de manière automatique.

```yaml
name: Create App directory
ansible.builtin.file:
  state: directory
  path: /var/www/ansible-app
  mode: 0755

- name: Copy template file
ansible.builtin.template:
  src: templates/db-config.php.j2
  dest: /var/www/ansible-app/db-config.php

- name: Copy app files
ansible.builtin.copy:
  src: files/
  dest: /var/www/ansible-app/


- name: Create custom image for apache and php
community.docker.docker_image:
  name: my-apache
  build:
    path: /var/www/ansible-app
  source: build

- name: Create container apache
community.docker.docker_container:
  name: web
  image: my-apache
  state: started
  restart_policy: on-failure
  ports:
    - "80:80"
  volumes:
  - /var/www/ansible-app:/var/www/html
```

Déploiement de Docker

```
PLAY [all] *********************************

TASK [Gathering Facts] *********************
ok: [node.2]
ok: [node.1]

TASK [docker : install prerequisites] ******
ok: [node.2]
ok: [node.1]

TASK [docker : add apt-key] ****************
ok: [node.2]
ok: [node.1]

TASK [docker : add docker repo] ************
ok: [node.2]
ok: [node.1]
```

Conclusion

Le projet était intéressant car on a pu exploiter les connaissances vues en cours mais également lire la doc qui est très utile pour savoir le fonctionnement et la syntaxe.

De plus on a eu quelques soucis d'infrastructure que ce soit au niveau des échanges des clés ssh pour autoriser les connexions pour chacun d'entre nous et pouvoir nous connecter sur l'infrastructure hébergé sur Digital Ocean.

Nous avons rencontré d'autres problèmes durant le projet, comme un souci de version ansible et python qui ne nous permettais pas de mener à bien le projet, on aura pu le résoudre en suivant.

Pour la partie BDD il y'a eu des soucis par rapport aux templates, il a donc fallu adapter le playbook.

Enfin il aurait été bien d'avoir un jalon retardé pour permettre de mener le projet jusqu'au bout autrement dit avec la partie reverse proxy.

Sources

https://docs.ansible.com/

https://dev.mysql.com/doc/

https://httpd.apache.org/docs/2.4/fr/

https://www.php.net/manual/fr/index.php

FIN DU DOCUMENT