**Documentation technique**

Dans notre projet, nous allons mettre en place une petite infrastructure contenant des conteneurs Docker afin d'exécuter une application. L'infrastructure comprendra un loadbalancer, deux serveurs web, un serveur de base de données et un réseau DMZ pour chaque serveur web.

**Énoncé**

* Le loadbalancer devra être mis en place avec l'outil HaProxy.
* Le réseau DMZ1 sera un réseau en 192.168.50.0/24.
* Le réseau DMZ2 sera un réseau en 10.0.50.0/24.
* Le serveur web1 et web2 hébergeront les applications, qui sont disponibles depuis Git Hub.
* Le serveur BDD contiendra la base de données.

**RESOLUTION DU SUJET**

Nous commencerons par installer Docker sur la VM. Pour ce faire, nous pouvons utiliser la commande suivante :

**sudo apt-get install docker**

**Configuration des réseaux DMZ**

Ici nous allons mettre en place et configurer les réseaux DMZ. Pour ce faire, nous utiliserons la commande suivante :

sudo docker network create dmz1

sudo docker network create dmz2

Configuration du loadbalancer

Ici nous allons mettre en place et configurer le loadbalancer. Nous utiliserons donc un fichier Dockerfile comme suit :

Dockerfile

FROM haproxy:1.8.0

ADD haproxy.cfg /usr/local/etc/haproxy/haproxy.cfg

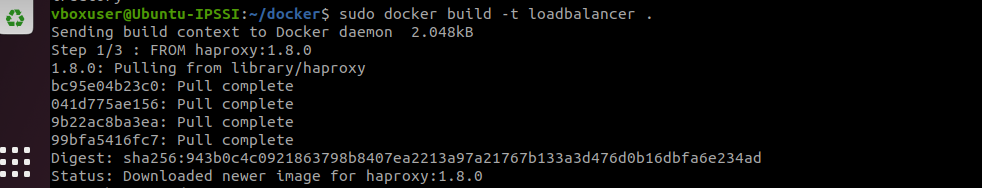
EXPOSE 80

Le fichier Dockerfile télécharge le fichier de configuration HaProxy depuis le dossier local et l'expose sur le port 80.

Pour construire et démarrer le loadbalancer, nous utiliserons les commandes suivantes :

docker build -t loadbalancer .

docker run -d --network dmz1 --name loadbalancer loadbalancer



Configuration des serveurs web

Nous allons maintenant configurer les serveurs web. Pour ce faire, nous utiliserons les fichiers Dockerfiles suivants :

Dockerfile

FROM nginx:1.20.1

COPY index.html /usr/share/nginx/html/

EXPOSE 80

Dockerfile

FROM nginx:1.20.1

COPY index.html /usr/share/nginx/html/

EXPOSE 80

Les fichiers Dockerfiles téléchargent la page d'accueil depuis le dossier local et l'exposent sur le port 80.

Pour construire et démarrer les serveurs web, nous utiliserons les commandes suivantes :

docker build -t web1 .

docker run -d --network dmz1 --name web1 web1

docker build -t web2 .

docker run -d --network dmz2 --name web2 web2

Configuration du serveur de base de données

Nous allons maintenant configurer le serveur de base de données. Pour ce faire, nous utiliserons le fichier Dockerfile suivant :

Dockerfile

FROM postgres:14.2

EXPOSE 5432

Le fichier Dockerfile expose la base de données sur le port 5432.

Pour construire et démarrer le serveur de base de données, nous utiliserons les commandes suivantes :

Bash

docker build -t db .

docker run -d --network dmz1 --name db db

Nous allons utiliser un script pour effectuer l’automatisation

Script d'automatisation

Le script d'automatisation est le suivant :

Bash

#!/bin/bash

# Récupération des nouveaux fichiers Dockerfiles et Docker compose

git clone https://github.com/landry1811/depodocker.git

# Construction des nouveaux conteneurs Docker

docker-compose build

# Démarrage des nouveaux conteneurs Docker

docker-compose up -d

Ce script clone le dépôt GitHub contenant les nouveaux fichiers Dockerfiles

