Guide Général: Configuration d'un Load Balancer avec Docker Compose

Introduction

Ce document fournit une méthode générale pour configurer un Load Balancer avec Docker Compose. Cette

configuration est applicable à plusieurs types de services backend, pas uniquement PHP. Le Load Balancer

distribue les requêtes entre plusieurs instances de services pour améliorer la disponibilité et l'équilibrage de

la charge.

1. Fichier docker-compose.yml

Le fichier docker-compose.yml définit plusieurs services backend et un service Load Balancer. Voici un

exemple:

version: '3.8'

services:

backend1:

build: ./backend

container\_name: backend1

expose:

- '80'

backend2:

build: ./backend

container\_name: backend2

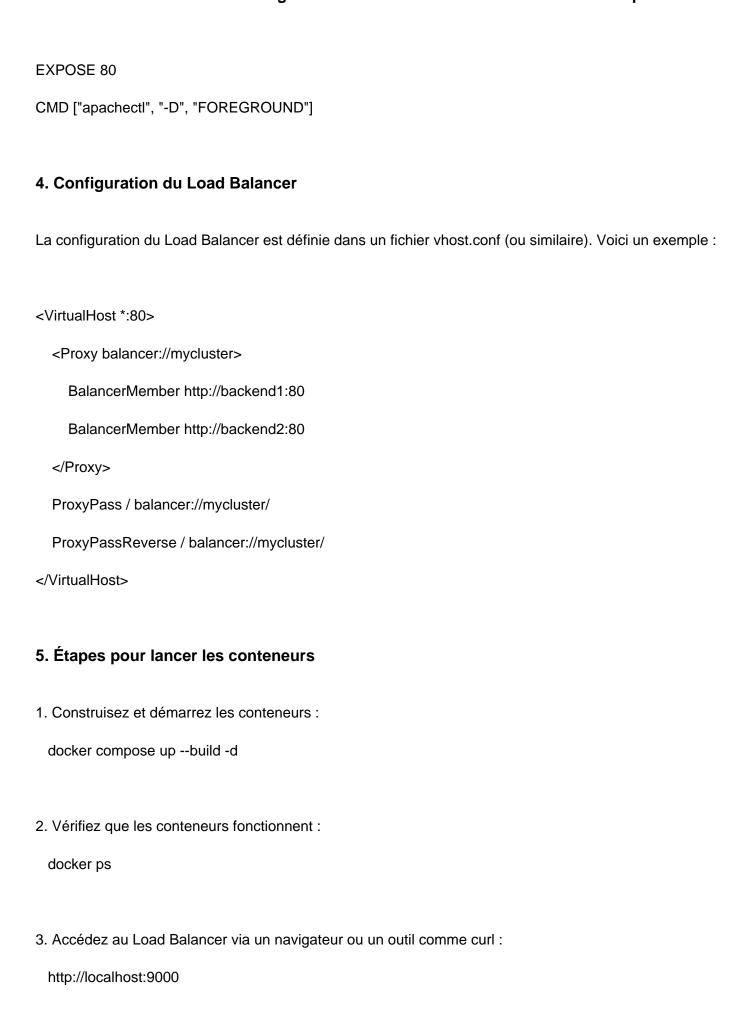
expose:

- '80'

## Guide Général : Configuration d'un Load Balancer avec Docker Compose

load-balancer:
build:
context: ./load-balancer
ports:
- '9000:80'
2. Dockerfile des services backend
Chaque service backend est défini dans un Dockerfile. Voici un exemple pour un backend générique :
FROM <image base,="" de="" exemple,="" node:14="" ou="" par="" php:7.4-apache=""/>
WORKDIR /usr/src/app
COPY
RUN <commande au="" construction="" de="" service="" spécifique=""></commande>
EXPOSE 80
CMD [" <commande au="" de="" démarrage="" service="" spécifique="">"]</commande>
3. Dockerfile du Load Balancer
Le Load Balancer est configuré avec Apache. Voici un exemple de Dockerfile :
FROM debian:latest
RUN apt-get update && apt-get install apache2 -y
RUN a2enmod proxy proxy_http proxy_balancer lbmethod_byrequests
COPY vhost.conf /etc/apache2/sites-available/000-default.conf

## Guide Général: Configuration d'un Load Balancer avec Docker Compose



## Guide Général : Configuration d'un Load Balancer avec Docker Compose

Le Load Balancer distribuera les requêtes entre les services backend.

## Conclusion

Cette méthode est générale et peut être adaptée à divers types de services backend (Node.js, PHP, Python, etc.). Le Load Balancer Apache peut être remplacé par un autre outil comme Nginx en modifiant la configuration. Cette approche garantit une répartition efficace de la charge entre plusieurs instances de service.