## Exercices 6 – Nginx

Informations

**Évaluation** : formative

**Type de travail** : individuel

**Durée** : 3 heures

**Système d’exploitation** : Ubuntu serveur

**Environnement** : virtuel

Objectifs

Cet exercice a pour objectifs :

* Faire une installation complète d’un site Web avec nginx.
* Utiliser nginx comme un équilibreur/répartiteur de charge (load balancer).

Dans cet exercice, nous allons nous créer un site Web complet comprenant le serveur Web nginx, php et MySQL. Nous allons aussi voir comment utiliser nginx comme équilibreur de charge (load balancer).

#### Partie 1 : Installation complète d’un site Web

Dans cette partie, vous allez installer le serveur Web nginx selon les spécifications données.

Étape 1 : Installation

1. Consulter le site Web <https://adoltech.com/blog/how-to-set-up-nginx-php-fpm-and-mysql-with-docker-compose/> pour comprendre une installation comme nous voulons faire.

Step 2

server {

server\_name docker-demo.com;

root /var/www/html;

index index.php index.html index.htm;

access\_log /var/log/nginx/access.log;

error\_log /var/log/nginx/error.log;

location / {

try\_files $uri $uri/ /index.php?$query\_string;

}

# PHP-FPM Configuration Nginx

location ~ \.php$ {

try\_files $uri = 404;

fastcgi\_split\_path\_info ^(.+\.php)(/.+)$;

fastcgi\_pass php:9000;

fastcgi\_index index.php;

include fastcgi\_params;

fastcgi\_param REQUEST\_URI $request\_uri;

fastcgi\_param SCRIPT\_FILENAME $document\_root$fastcgi\_script\_name;

fastcgi\_param PATH\_INFO $fastcgi\_path\_info;

}

}

Step 3

version: '3'

services:

# Nginx server configuration

web:

image: nginx:1.17

container\_name: webserver

depends\_on:

- docker-mysql

- php

links:

- docker-mysql

- php

volumes:

- ./webcontent:/var/www/html

- ./nginx:/etc/nginx/conf.d/

ports:

- "8080:80"

# Php-fpm configuration

php:

image: php:7.2-fpm

volumes:

- ./webcontent:/var/www/html

- ./php:/usr/local/etc/php/php.ini

# Mysql configuration

docker-mysql:

image: mysql:5.7

container\_name: docker-mysql-demo

restart: always

environment:

- MYSQL\_DATABASE=docker\_demo

- MYSQL\_ROOT\_PASSWORD=root123

- MYSQL\_USER=root

- MYSQL\_PASSWORD=root123

ports:

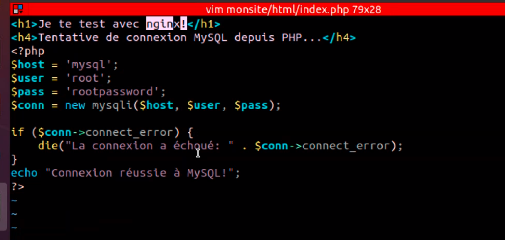
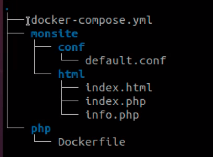
- "3306:3306"

volumes:

- /dbdata:/var/lib/mysql

1. Avec ce que vous avez lu en a, l’information de l’exercice 4 et l’information de l’exercice 5, créer un site Web avec nginx, php et MySQL avec les spécifications et la structure qui suit.

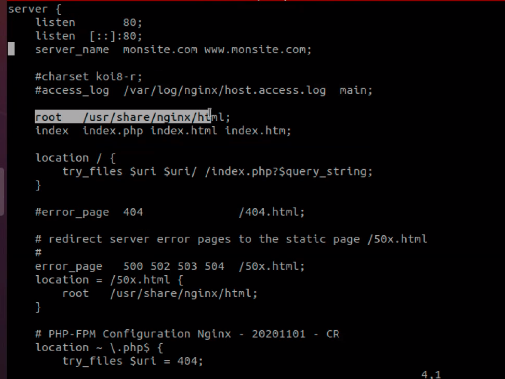
**Fichiers à utiliser** : utiliser le fichier index.php créer à l’exercice 4 pour tester votre site Web. (php et mySql)

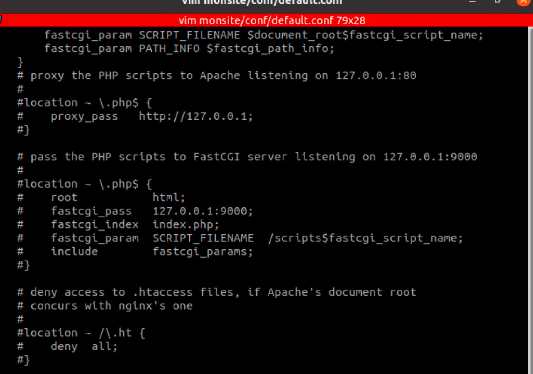


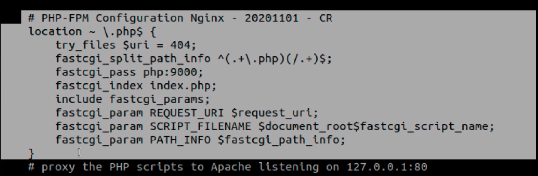
**Nom du site Web** : www.monsite.com, mais il doit également répondre à monsite.com. il faut inclure le index.php (fait dans le host)

**Pour nginx** : utiliser l’image nginx:alpine, utiliser le port 80, ne changer pas les fichiers journaux (logs) et l’emplacement de la racine (/) du site.

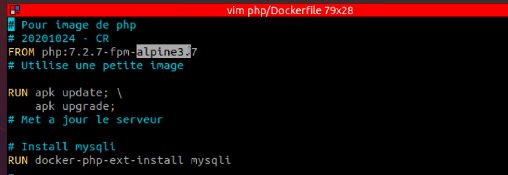
**(step 2)**





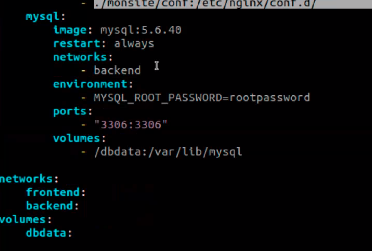


**Pour php** : on n’utilise pas de fichier php.ini, seulement le Dockerfile pour mettre à jour php et installer mysqli.



Dans le Docker-compose





/SiteWeb/

|--docker-compose.yml

|--monsite/

| |--conf/

| |--site.conf

| |--html/

| |--index.php # celui de l’exercice 4, ajuster les paramètres si nécessaire

|--php/

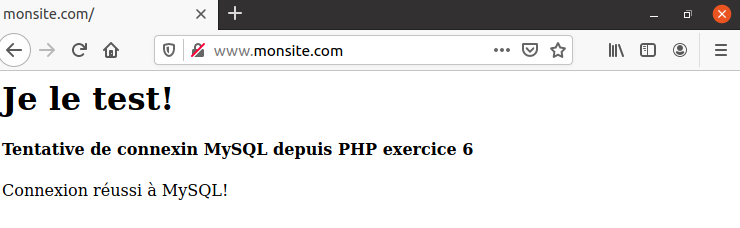
|-- Dockerfile # pour php, prenez celui de l’exercice 4

1. Tester votre site.

docker-compose up –build -d

Sur le browser  monsite.com

Docker-compose down –rmi local



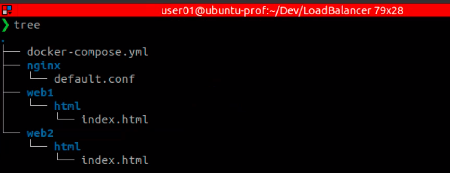
#### Partie 2 : Load Balancer

Dans cette partie, vous allez configurer nginx comme équilibreur de charge entre deux serveurs.

En plus d'être un serveur Web et de cache, NGINX est également un équilibreur de charge. Son style de configuration est flexible, mais quelque peu non évident.

Étape 1 : Création de la structure

1. Sous Dev, créer la structure suivante :



/LoadBalancer/

├── docker-compose.yml

├── nginx/

│   └── default.conf # ce fichier est vide pour l’instant (voir étape 2)

├── web1/

│   └── html/

│   └── index.html

└── web2/

└── html/

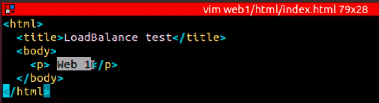
└── index.html

Nous avons nginx qui va rediriger le trafic pour www.monsite.com à deux serveurs Apache (httpd) en round robin. Donc, nous allons avoir cela.

|-- web1 (httpd)

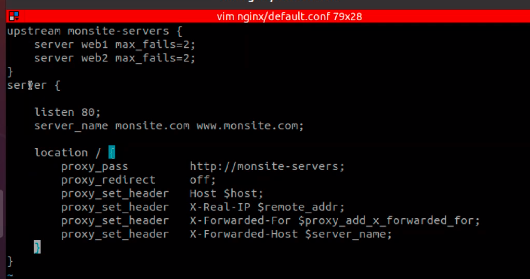
www.monsite.com── nginx|

   |-- web2 (httpd)



1. Pour pouvoir voir la répartition entre les deux serveurs Apache, créez deux pages index.html différentes pour les deux serveurs.

Étape 2 : Configuration



1. Nous allons utiliser le module upstream pour créer un groupe nommé de serveurs. Par exemple, la clause suivante définit les serveurs monsite-servers comme une collection de deux serveurs.

upstream monsite-servers {

server web1 max\_fails=2;

server web2 max\_fails=2;

}

// C’est le nom du container

Cette clause doit se retrouver dans le contexte http. Nous savons, voire l’exercice 5, que le fichier default.conf est inclus dans le contexte http. Vous pouvez donc mettre cette clause dans le fichier default.conf.

1. Les groupes upstream peuvent être référencés à partir de définitions d'hôte virtuel. En particulier, ils peuvent être utilisés comme destinations de proxy, tout comme les noms d'hôte. Dans le contexte server, qu’y va se retrouver dans le fichier default.conf, vous allez faire une configuration pour que le trafic pour monsite.com et www.monsite.com soit détourné vers les serveurs web1 et web2 en round robin (par défaut).

server {

listen 80;

server\_name monsite.com www.monsite.com;

location / {

proxy\_pass http://monsite-servers;

proxy\_redirect off;

proxy\_set\_header Host $host;

proxy\_set\_header X-Real-IP $remote\_addr;

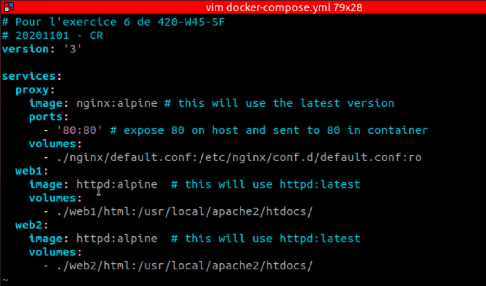
proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Host $server\_name;

}

}

1. En utilisant les références à la fin du document, identifier la signification des différentes directives proxy\_.
2. Maintenant, créer le fichier docker-compose.yml pour créer vos conteneurs.



version: '3'

services:

//nginx

proxy:

image: nginx:alpine # utilise la version la plus récente

ports:

- '80:80' # expose 80 sur l’hôte et envoie à 80 dans conteneur

volumes:

- ./nginx/default.conf:/etc/nginx/conf.d/default.conf:ro

// 1 serveur apache

web1:

image: httpd:alpine # utilise la version httpd:lapine

volumes:

- ./web1/html:/usr/local/apache2/htdocs/

// 2e serveur apache

web2:

image: httpd:alpine # utilise la version httpd:alpine

volumes:

- ./web2/html:/usr/local/apache2/htdocs/

1. Lancer vos conteneurs et vérifier que l’équilibrage de charge fonctionne entre les deux serveurs. Rafraîchir la page et vous devriez voir le changement entre les deux pages index.html.

On va alterner entre Web1, Web2

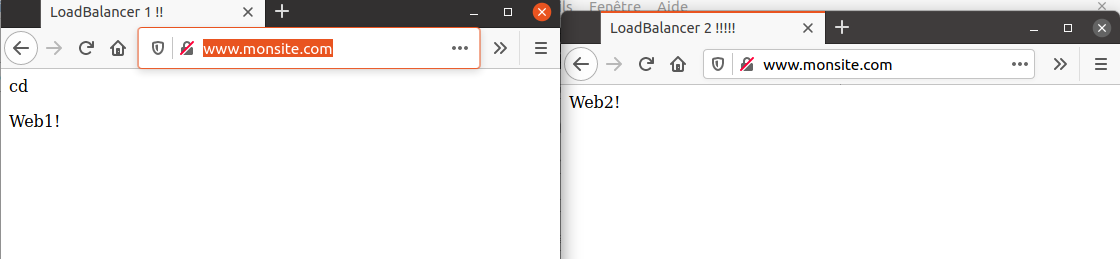
1. Comment le lien se fait-il entre les conteneurs httpd et la clause upstream ?

Rép : 2

1. Arrêter et effacer vos conteneurs.

Pour vérification

Remettre une capture d’écran de votre Linux avec Firefox ayant deux fenêtres ouvertes sur www.monsite.com montrant les deux retours différents.



Compétences développées

|  |  |
| --- | --- |
| **00SJ** – Effectuer le déploiement de serveurs intranet. | **00SJ # 1** – Analyser le projet de déploiement.  **00SJ # 2** – Monter les serveurs intranet.  **00SJ # 3** – Installer les services intranet.  **00SJ # 5** – Participer à la mise en service des serveurs intranet. |

**Note** : les compétences sont développées en partie.

Références

<https://nginx.org/en/docs/>

<https://docs.nginx.com/nginx/admin-guide/>

<https://nginx.org/en/docs/http/load_balancing.html>

<https://nginx.org/en/docs/http/ngx_http_proxy_module.html>

<https://hub.docker.com/_/nginx>

<https://adoltech.com/blog/how-to-set-up-nginx-php-fpm-and-mysql-with-docker-compose/>