## Exercice 7 – Nginx (Corrigé)

## Remarque : La couleur de police rouge ou les surlignements gris indiquent le texte qui apparaît uniquement dans la copie corrigée.

Informations

**Évaluation** : formative

**Type de travail** : individuel

**Durée** : 3 heures

**Système d’exploitation** : Ubuntu serveur

**Environnement** : Docker

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.

Dans ce document, vous regardez la configuration du fichier site.conf et l’adapter à votre site.

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 ou MariaDB 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.

**Nom du site Web** : www.monsite.com, mais il doit également répondre à monsite.com.

**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. Faire un mappage du répertoire monsite/conf au répertoire /etc/nginx/conf.d.

**Pour php** : on n’utilise pas de fichier php.ini comme dans l’exemple plus haut. Vous créez seulement un Dockerfile pour installer mysqli.

**Pour MySQL/MariaDB**: assurez-vous que le conteneur reparte s’il arrête involontairement et créer un volume persistant dbdata.

/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.

#### 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

├── 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. Pour nginx, 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;

}

Cette clause doit se retrouver dans le contexte http. Nous savons, voire l’exercice 6, 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:

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

web1:

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

volumes:

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

web2:

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

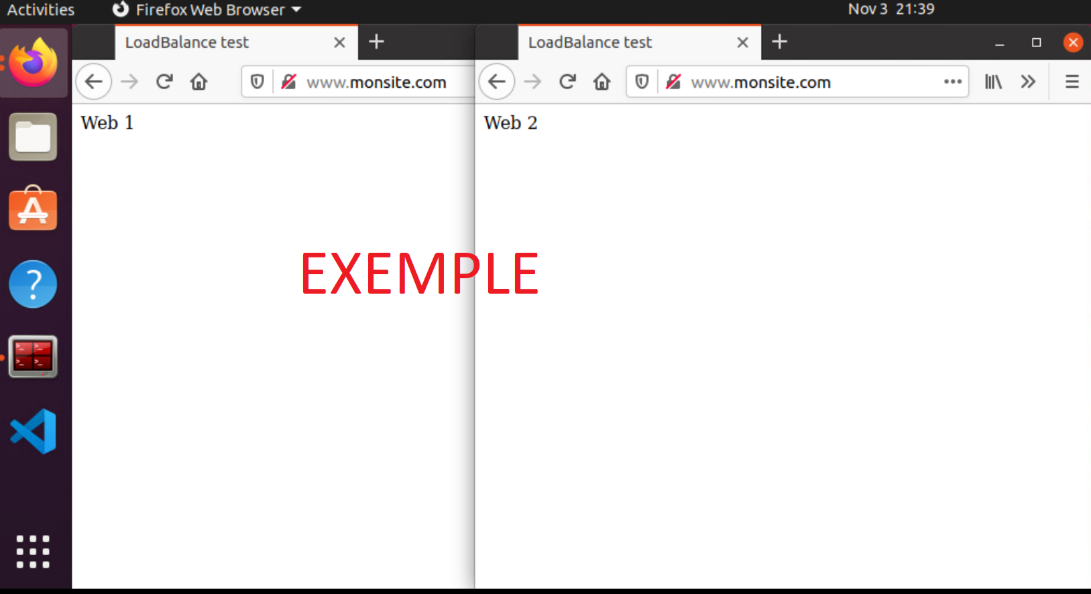
volumes:

- ./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.
2. Comment le lien se fait-il entre les conteneurs httpd et la clause upstream ? Les noms des conteneurs sont définis dans le « DNS » de docker. C’est pourquoi que nous référençons web1 et web2 dans la clause upstream.
3. 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/>

Solution partie 6.1

**# docker-compose.yml**

version: "3.2"

services:

php:

build: './php/'

networks:

- backend

volumes:

- ./monsite/html:/usr/share/nginx/html

nginx:

image: nginx:alpine

depends\_on:

- php

- mariadb

networks:

- frontend

- backend

ports:

- "80:80"

- "443:443"

volumes:

- ./monsite/html:/usr/share/nginx/html

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

mariadb:

image: mariadb

restart: always

networks:

- backend

environment:

- MYSQL\_ROOT\_PASSWORD=rootpassword

ports:

- "3306:3306"

volumes:

- dbdata:/var/lib/mysql

networks:

frontend:

backend:

volumes:

dbdata:

**# monsite default.conf**

server {

listen 80;

listen [::]:80;

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

#charset koi8-r;

#access\_log /var/log/nginx/host.access.log main;

root /usr/share/nginx/html;

index index.php index.html index.htm;

location / {

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

}

#error\_page 404 /404.html;

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

# PHP-FPM Configuration Nginx - 20201101 - CR

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;

}

# proxy the PHP scripts to Apache listening on 127.0.0.1:80

#

#location ~ \.php$ {

# proxy\_pass http://127.0.0.1;

#}

# pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000

#

#location ~ \.php$ {

# root html;

# fastcgi\_pass 127.0.0.1:9000;

# fastcgi\_index index.php;

# fastcgi\_param SCRIPT\_FILENAME /scripts$fastcgi\_script\_name;

# include fastcgi\_params;

#}

# deny access to .htaccess files, if Apache's document root

# concurs with nginx's one

#

#location ~ /\.ht {

# deny all;

#}

}

**# php Dockerfile**

# Pour image de php

FROM php:7.2.7-fpm-alpine3.7

# Utilise une petite image

RUN apk update; \

apk upgrade;

# Met a jour le serveur

# Install mysqli

RUN docker-php-ext-install mysqli

**# index.php**

<h1>Je te test!</h1>

<h4>Tentative de connexion MariaDB depuis PHP...</h4>

<?php

$host = 'mariadb';

$user = 'root';

$pass = 'rootpassword';

$conn = new mysqli($host, $user, $pass);

if ($conn->connect\_error) {

die("La connexion a échoué: " . $conn->connect\_error);

}

echo "Connexion réussie à MariaDB!";

?>

Solution partie 6.2

**# docker-compose.yml**

# Pour l'exercice 6 de 420-W45-SF

# 20201101 - CR

version: '3'

services:

proxy:

image: nginx:alpine # this will use the latest version

ports:

- '80:80' # expose 80 on host and sent to 80 in container

volumes:

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

web1:

image: httpd:alpine # this will use httpd:latest

volumes:

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

web2:

image: httpd:alpine # this will use httpd:latest

volumes:

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

**# nginx default.conf**

upstream monsite-servers {

server web1 max\_fails=2;

server web2 max\_fails=2;

}

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;

}

}