## Exercice 5 – Apache, installation d’un site Web et site virtuel

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.
* Installer des sites Web virtuels.

Dans cet exercice, vous allez compléter l’installation de l’exercice 3 pour ajouter un logiciel de base de données à votre installation. Vous allez également créer différents hôtes sur le même serveur à l’aide du principe d’hôtes virtuels.

#### Partie 1 : Site Web complet

Dans cette partie, vous allez compléter votre installation de site Web en incluant le logiciel de base de données MariaDB. Veuillez noter que certains principes de sécurité ont été ignorés ici pour des raisons de simplicité et de démonstration. À savoir PHP utilise l'identifiant root, le mot de passe MariaDB codé en « dur » et faible, pour n'en nommer que quelques-uns! Ce code n’est pas fait pour être exécuté en production!

Étape 1 : Préparer la structure de répertoire

1. Je vous rappelle la structure de répertoire que nous avons.

/Sites/

|--docker-compose.yml

|--monsite/

| |--Dockerfile

| |--certs

| |--server.crt

| |--server.key

| |--conf/

| |--httpd.conf

| | |--http-ssl.conf

| |--html/

| |--index.html

| |--index.php

|--php/

|-- Dockerfile

1. Pour pouvoir vérifier php avec MariaDB, nous allons éditer le fichier index.php dans le répertoire monsite/html avec le code suivant :

<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!";

?>

Étape 2 : PHP et MariaDB avec Apache

1. Modifier votre fichier docker-compose.yml pour inclure MariaDB.

version: "3.2"

services:

php:

build: './php/'

networks:

- backend

volumes:

- ./monsite/html:/srv/htdocs

**apache**:

build: './monsite/'

depends\_on:

- php

**- mariadb**

networks:

- frontend

- backend

ports:

- "8080:80"

- "443:443"

volumes:

- ./monsite/html:/srv/htdocs

**mariadb:**

**image: mariadb**

**networks:**

**- backend**

**environment:**

**- MYSQL\_ROOT\_PASSWORD=rootpassword**

networks:

frontend:

backend:

Vous pouvez voir qu’il nous manque un volume pour la base de données, car dans cet exemple nous n’avons pas de base de données.

1. Modifier le Dockerfile de php pour inclure mysqli.

# Pour image de php

FROM php:fpm-alpine

# Utilise une petite image

RUN apk update; \

apk upgrade;

# Met a jour le serveur

**# Install mysqli**

**RUN docker-php-ext-install mysqli**

1. Lancez vos conteneurs et allez à la page https://www.monsite.com ou https://www.monsite.com/index.php. Il se peut que la connexion échoue à la première tentative.

Est-ce que votre site Web fonctionne avec MariaDB ?

1. Arrêter vos conteneurs.

#### Partie 2 : Site Web virtuel

Apache prend en charge la création de sites Web séparés au sein d'un même serveur et permet de garder le contenu séparé. Les sites individuels sont configurés sur le même serveur dans ce que l'on appelle des hôtes virtuels.

Le terme hôte virtuel fait référence à la pratique consistant à exécuter plusieurs sites Web (tels que societe1.exemple.com et societe2.exemple.com) sur une seule machine. Les hôtes virtuels peuvent être «basés sur IP», ce qui signifie que vous avez une adresse IP différente pour chaque site Web, ou «basés sur le nom», ce qui signifie que vous avez plusieurs noms en cours d'exécution sur chaque adresse IP. Le fait qu'ils s'exécutent sur le même serveur physique n'est pas perçu par l'utilisateur final.

Nous allons nous concentrer sur les hôtes virtuels «basés sur le nom».

Étape 1 : Site par défaut

1. Créer un répertoire vSites sous Dev et créer la structure de répertoire suivante :

/vSites/

|--conf/

|--site01/

| |--html/

|--site02/

| |--html/

1. Copier votre fichier index.html créer dans l’exercice 2 dans le répertoire html de site01 et site02. Modifier le fichier pour pouvoir identifier le site.
2. Créer un Dockerfile dans vSites pour créer une image httpd:latest. Dans l’image, mettre à jour le serveur, et exposer les ports 80 et 443.
3. Créer un fichier docker-compose.yml pour construire et lancer le conteneur httpd:latest que vous allez appeler apache, exposer le port 80 au port 80 du conteneur et créer un point de montage du répertoire site01/html vers le répertoire /usr/local/apache2/htdocs du conteneur.
4. Modifier le fichier hosts de votre VM pour ajouter www.site01.com à 127.0.0.1.
5. Lancer votre système avec docker-compose. Vérifier que ça fonctionne avec Firefox.
6. Ouvrez un shell sur votre conteneur apache et récupérez le fichier de configuration httpd.conf.

Copiez le ficher httpd.conf dans le répertoire /usr/local/apache2/htdocs retourner dans votre VM Ubuntu et vous aller apercevoir le nouveau fichier dans le répertoire site01/html :).

Déplacer le fichier dans le répertoire vSites/conf.

1. Arrêter votre système.
2. Nous allons modifier le fichier vSites/conf/httpd.conf pour pouvoir inclure le fichier httpd-vhosts.conf.

# Inclure le fichier http-vhosts.conf

# Elever le commentaire devant la ligne suivante

Include conf/extra/httpd-vhosts.conf

1. Copier le ficher httpd-vhosts.conf récupérer de l’exercice 2 dans vSites/conf.

Lorsque vous ajoutez un hôte virtuel basé sur le nom (namebased) à un serveur existant et que les arguments d'hôte virtuel correspondent à des combinaisons IP et de port préexistant, les demandes seront désormais traitées par un hôte virtuel explicite. Dans ce cas, il est généralement judicieux de créer un hôte virtuel par défaut avec un ServerName correspondant à celui du serveur de base. Les hôtes virtuels basés sur le nom pour l'ensemble des <virtualhost> sont traités dans l'ordre dans lequel ils apparaissent dans la configuration. Le premier ServerName ou ServerAlias correspondant est utilisé, sans priorité différente pour les caractères génériques (ni pour ServerName et ServerAlias).

Donc, le premier conteneur <VirtualHost \*:80> dans le fichier httpd-vhosts.conf devient notre serveur par défaut. Modifier le fichier httpd-vhosts.conf pour que le premier conteneur corresponde à notre serveur actuel (vous pouvez mettre le conteneur du deuxième site Web dummy-host2.example.com en commentaire).

<VirtualHost \*:80>

ServerAdmin clroy@csfoy.ca

DocumentRoot "/usr/local/apache2/docs/site01.com"

ServerName site01.com

ServerAlias www.site01.com

ErrorLog "logs/site01.com-error\_log"

CustomLog "logs/site01.com-access\_log" common

</VirtualHost>

1. Modifier le fichier Dockerfile pour ajouter le nouveau répertoire et copier les fichiers httpd.conf et httpd-vhosts.conf. Modifier le fichier docker-compose.yml pour tenir compte des changements du fichier httpd-vhosts.conf. Ne pas tenir compte des logs pour l’instant.

**Attention** : vous devez utiliser le paramètre -p avec mkdir pour créer un répertoire ayant un sous-répertoire.

1. Lancer votre système.
2. Allez à la page du site sur votre navigateur.

Avez-vous la page du site01 ?

Si oui, faire une modification au fichier site01/html/index.html et rafraîchir la page dans Firefox pour voir le changement.

Le problème, nous avons créé un nouveau répertoire, mais nous n’avons pas donné de permission à ce répertoire.

1. Vous pouvez arrêter votre système.
2. Ouvrer httpd.conf et regarder les permissions de /usr/local/apache2/htdocs. Dans le conteneur du serveur site01, du fichier httpd-vhosts.conf, ajouter un conteneur <Directory> pour donner les permissions d’accès au répertoire de site01.
3. Lancer votre conteneur.
4. Allez à la page du site sur votre navigateur.

Avez-vous la page du site01 ?

Étape 2 : Logs

1. Regarder les logs avec docker-compose (docker-compose logs).

Voyez-vous les logs de httpd (apache) ?

Voyez-vous les logs d’accès au site site01 (vous pouvez rafraîchir la page de votre navigateur pour générer des logs d’accès) ?

1. Les logs d’accès du site01 ne sont pas dirigés vers docker, mais vers le fichier logs/site01.com-access\_log. Faites les modifications nécessaires pour que les logs, erreurs et accès, se retrouvent dans le dossier /var/log/apache2 de votre VM.
2. Vérifier que les logs sont bien dans le répertoire de votre VM (si ça ne fonctionne pas, changer les droits du répertoire /var/log/apache2/ avec la commande chmod).

Étape 3 : Deuxième site

1. Vous allez maintenant ajouter le deuxième site, site02.com. Voici les spécifications pour le site :

**Nom de domaine** : le site doit être accessible sur votre VM/hôte avec site02.com et www.site02.com.

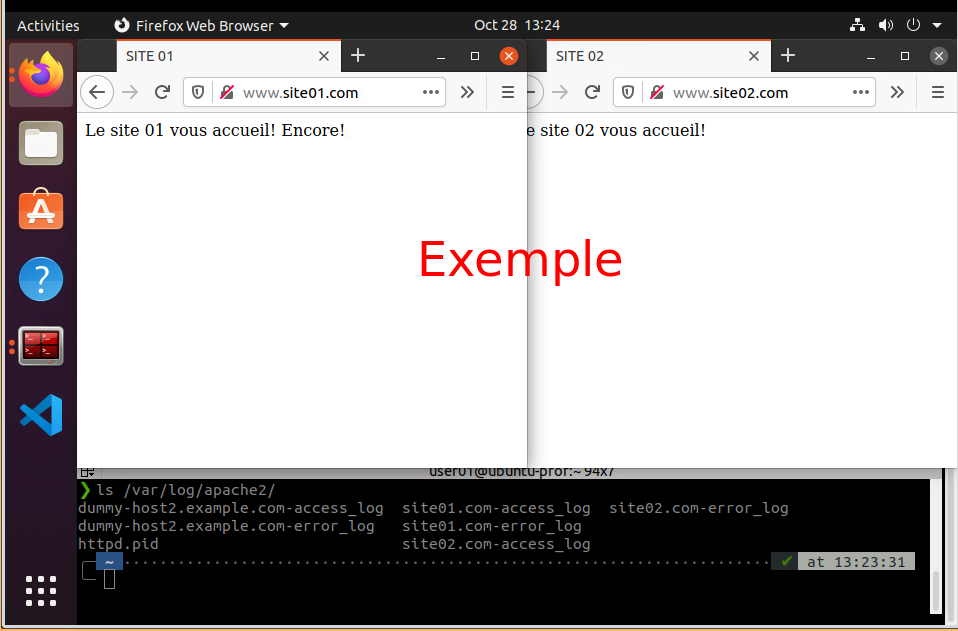
**Répertoire des documents** : /usr/local/apache2/docs/site02.com.

**Les logs** : dans /var/log/apache2 de votre VM.

1. Tester que tout fonctionne.

Pour vérification

Remettre une capture d’écran de votre Linux avec Firefox ouvert sur les deux sites de votre serveur Web et un shell qui affiche le répertoire /var/log/apache2.



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://httpd.apache.org/docs/2.4/>

<https://httpd.apache.org/docs/2.4/vhosts/>

<https://httpd.apache.org/docs/2.4/vhosts/name-based.html>

<https://cwiki.apache.org/confluence/display/httpd/DistrosDefaultLayout>

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

<https://docs.docker.com/compose/compose-file/>

<https://docs.docker.com/engine/reference/builder/>