

**TP N°11 Bis - Orchestration de conteneurs multi-services avec Docker Compose :  
déploiement des applications GLPI et OCS Inventory****Durée : 2 séances de 2h00**

| Moyens mis à disposition des étudiants                                                                                                                       |                                                                                                                                                                                                                                                                                                          |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Matériels :</b> <ul style="list-style-type: none"><li>• 1 poste de travail sous Windows 10 ou 11 professionnel</li><li>• Switch cisco SG 300-10</li></ul> | <b>Logiciels et systèmes d'exploitation :</b> <ul style="list-style-type: none"><li>• VirtualBox 7.2.6 et son pack d'extension</li><li>• Debian Bookworm 12.11</li><li>• Docker Engine 29.3</li><li>• Docker Compose 5.1.1</li><li>• Contenaïrd 2.2.2</li><li>• Navigateur Internet (au choix)</li></ul> |
| <b>Divers :</b> Accès à Internet                                                                                                                             |                                                                                                                                                                                                                                                                                                          |

| Savoir-faire mis en œuvre                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>❖ Justifier le choix d'une solution de mise en production d'un système</li><li>❖ Justifier le choix d'une solution de la disponibilité d'un serveur</li><li>❖ Installer et configurer une solution de disponibilité d'un serveur</li><li>❖ Sécuriser un serveur et une solution technique d'accès</li><li>❖ Administrer un système</li><li>❖ Valider et documenter une solution</li></ul> |

**Conditions de travail :**

- Individuel
- **Le compte-rendu de ce TP sera rédigé sous la forme d'une documentation technique (page de garde + sommaire + bibliographie et/ou sitographie)** présentant les étapes d'installation, de configuration, de paramétrage ainsi qu'une phase de recette (les tests) des services déployés via Linux Debian Bookworm.
- L'infrastructure que vous devez mettre en place est présentée en annexes **document 1**.

**Contexte professionnel : CHU de Pointe-à-Pitre**

Etablissement de référence dans l'organisation du système de santé guadeloupéen, le CHU de Pointe-à-Pitre assure les missions qui lui ont été confiées par l'Etat et notamment les soins, la prévention, l'enseignement et la recherche. Il est doté d'un conseil de surveillance et dirigé par un directeur général assisté d'un directoire.

Le CHU c'est près de 100 corps de métiers différents, assurant l'emploi de 2971 agents, 331 médecins et internes répartis dans les différents départements et directions suivants :

- Direction des affaires médicales
- Direction de la recherche
- Direction des ressources humaines et des relations sociales
- Direction de la formation et des conditions de travail
- Direction des travaux, des services techniques et de la sécurité
- Direction des services économiques, logistiques et biomédicaux, ...

Le département Informatique et du système d'information est subdivisé en services. Nous retrouvons les services :

- système et exploitation ;
- réseaux et télécoms ;
- sécurité du système d'information ;
- étude et projets,
- et micro-informatique.

Le service « **micro-informatique** » comprend 3 techniciens informatiques. Ils sont chargés, au quotidien :

- d'assister les employés (utilisateurs) dans l'utilisation des dispositifs matériels et logiciels mis à leur disposition,
- d'identifier, de traiter et de résoudre les problèmes (incidents) rencontrés par les employés,
- de proposer et de mettre en œuvre toutes les solutions matérielles et logicielles permettant d'anticiper tous les problèmes (incidents) matériels, logiciels, systèmes et applicatifs pouvant se présenter sur le parc informatique...

Les techniciens du service « **micro-informatique** » en charge du parc informatique du CHU de Pointe-à-Pitre, ont décidé de déployer les applications « métiers » suivantes au sein de différents conteneurs afin de gérer le parc informatique, les configurations logicielles et matérielles mais aussi les incidents rencontrés par les personnels :

- ils ont retenu le **logiciel libre de gestion de parc informatique et de Helpdesk - GLPI** afin de détecter, d'enregistrer, de classer et de faire le suivi des incidents jusqu'à leur clôture.
- ils vont associer GLPI à un autre **logiciel libre OCS Inventory NG** afin de répertorier et faire l'inventaire des **dispositifs matériels et logiciels** disponibles sur leur parc informatique.

## 1. Présentation générale de Docker Compose

Dans le TP n°11 précédent, nous avons pu découvrir la puissance de Docker et les subtilités pour communiquer avec les conteneurs.

Cependant, les deux inconvénients des commandes Docker sont :

- la longueur et la complexité d'une seule ligne.
- l'impossibilité de lancer plusieurs conteneurs en une seule fois

Docker Compose est donc un outil qui va permettre de définir (de configurer) et d'exécuter des applications Docker multi-conteneurs, à partir d'une commande simple. Pour cela, Docker Compose s'appuie sur un fichier au format **YAML** contenant toute les configurations nécessaires pour les différents services à déployer dans chaque conteneur.

**L'intérêt est alors de ne lancer qu'une seule commande pour créer et démarrer tous les services associés aux conteneurs selon les configurations du fichier **YAML**.**

Un fichier **YAML (YAML Ain't a Markup Language)** est un format de représentation des données hiérarchique et lisible, basé principalement sur des paires "clé : valeur" et une structure hiérarchique par indentation (comme avec Python).

### Observations :

- Avec Docker Compose, les **services définis dans un même fichier sont automatiquement placés sur un réseau commun et peuvent se découvrir par leur nom de service.**
- Il y a une chose importante lorsque vous éditez un fichier **YAML**, ***l'indentation doit être faite avec un ou plusieurs espaces, mais jamais avec les tabulations.***

## 2. Syntaxe d'un fichier « docker-compose.yml »

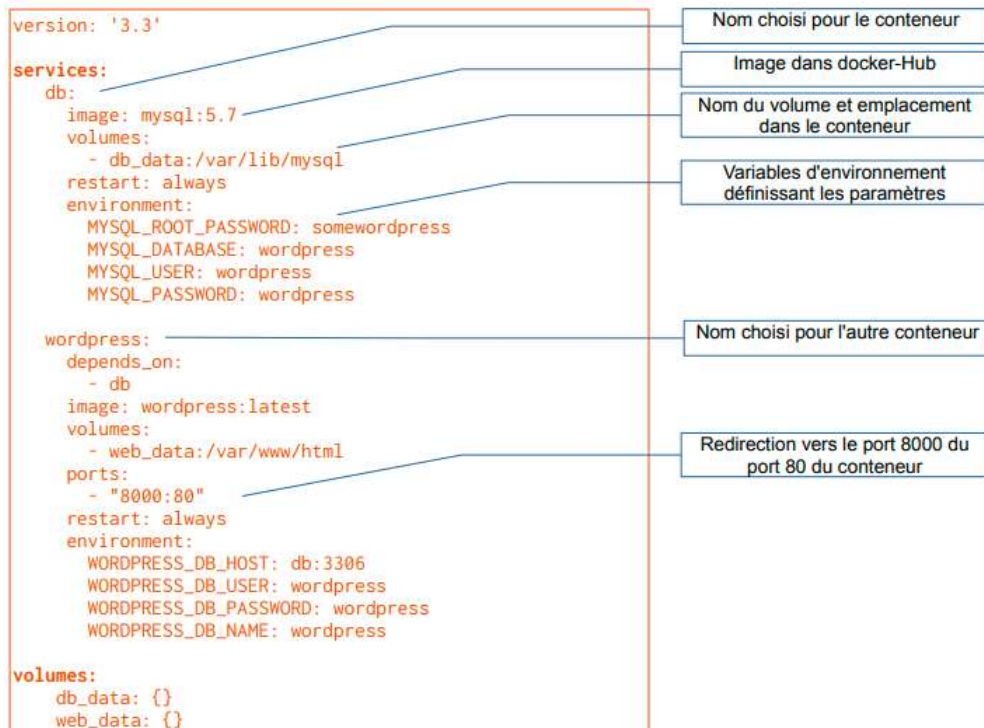
La commande « **docker compose up -d** » va lire et exécuter le contenu du fichier « **docker-compose.yml** ».

Connaître les éléments qui constituent ce fichier facilitera sa lecture/compréhension et permettra d'en créer autant que nécessaire en fonction des besoins.

Il existe de nombreuses instructions, mais voici celles qui sont les plus importantes :

- Le mot-clé « **service** » : permet de définir les différents conteneurs qui seront activés avec leur configuration.
  - L'usage du mot « **environment** » : permet de définir des variables d'environnement dans le conteneur.
- L'instruction « **network** » : permet de définir un **réseau** Docker est un espace de communication isolé où les conteneurs peuvent s'échanger des données. C'est comme un réseau local virtuel : les conteneurs connectés au même réseau peuvent se « voir », tandis que ceux sur des réseaux différents sont isolés.
- L'instruction « **volume** » : met en œuvre le mécanisme de persistance des données (voir TP n°11).

Voici un exemple de fichier « **docker-compose.yml** » permettant de démarrer deux conteneurs :



**Remarque :** le résultat obtenu suite à cette orchestration de conteneurs s'appelle une « **stack** ». Une **stack** est un ensemble d'outils ou de technologies qui fonctionnent ensemble.

**Exemple :** Dans le développement web, une **stack** est un groupe de technologies utilisées pour créer un site ou une application web :

- système d'exploitation (**Linux**)
- serveur web (**Apache**)
- base de données (**MySQL** ou **MariaDB**)
- langage backend (**PHP**)

Cette solution est qualifiée de « **stack** » **LAMP**.

### 3. Décomposition du contenu d'un fichier « docker-compose.yml » en ligne de commandes Docker

Il est donc possible de décomposer le contenu d'un fichier « **docker-compose.yml** » en ligne de commandes Docker par exemple :

| Docker                                       | Docker-compose.yml             |
|----------------------------------------------|--------------------------------|
| <code>docker run -d</code>                   | Version: '3'                   |
| <code>--name mysqldb</code>                  | service:                       |
| <code>-p 3307:3306</code>                    | mysqldb:                       |
| <code>-e MYSQL_ROOT_PASSWORD=password</code> | image: <b>mysql</b>            |
| <b>mysql</b>                                 | port:                          |
|                                              | - 3307:3306                    |
|                                              | environment:                   |
|                                              | - MYSQL_ROOT_PASSWORD=password |

**MISSION :** **Monsieur BELLON** est le Responsable du service « **micro-informatique** ». Vous êtes stagiaire au sein de son service et il vous demande d'orchestrer et de documenter le déploiement des services web applicatifs précités dans des conteneurs à l'aide de Docker Compose.

Dans un premier temps, vous devrez déployer les services et applicatifs suivants au sein de différents conteneurs afin d'appréhender le principe de fonctionnement de Docker Compose et vous familiariser avec ses 5 principales commandes, à maîtriser :

- **MySQL ou MariaDB** : système de gestion de BDD relationnelles permettant de stocker, organiser et gérer les données.
- **PhpMyAdmin** : application web offrant une interface graphique pour gérer une BDD relationnelle (souvent basée sous MariaDB ou MySQL).
- **WordPress** : est un système de gestion de contenu (CMS) pour créer et gérer un site web facilement sans pour autant l'avoir codé.

Dans un deuxième temps, vous devrez déployer SEUL les applications **PhpMyAdmin**, **GLPI** et **OCS Inventory** dans des conteneurs différents avec des bases de données gérées à partir du SGBDR **MySQL/MariaDB**, installé dans son propre conteneur.

#### Remarques :

- Soignez la présentation de votre compte-rendu.
- Les mots de passe qui seront utilisés lors de la configuration des services et applicatifs sont « **root → admin2526** » et « **tepson → tepson2526** », soyez donc vigilant sur ce point.
- Dans le tableau ci-dessous sont précisés les profils, identifiants et mots de passe des comptes utilisateurs qui seront créés sur les applications GLPI/OCS Inventory :

| NOM – Prénom      | Identifiant | Mot de passe  | Profil           |
|-------------------|-------------|---------------|------------------|
| Mathieu BELLON    | mbellon     | @mbellon2526@ | Admin/technicien |
| Yohann LEMARC     | ylemarc     | @ylemarc2526@ | Utilisateur      |
| Chantal DUPONT    | cdupont     | @cdupont2526@ | Utilisateur      |
| Christopher LOUIS | clouis      | @clouis2526@  | Utilisateur      |

Différents documents sont mis à votre disposition en annexes pour vous aider à appréhender Docker Compose :

**Document 1 : Architecture réseau à réaliser**

**Document 2 : Lancement d'une stack avec Docker Compose**

**Document 3 : Commandes essentielles de Docker Compose**

#### Travail à faire :

#### Etape 0 : Création de la VM « SRVDOCKER » et installation des composants logiciels de Docker

1. Créez ou clonez une machine virtuelle sous Linux Debian nommée « **srvdocker** » (*sans interface graphique*), selon la configuration suivante :

|                     |                       |
|---------------------|-----------------------|
| <b>Nom</b>          | srvdocker             |
| <b>Disque dur</b>   | 20 Go contrôleur SATA |
| <b>Mémoire</b>      | 4 Go                  |
| <b>CD/DVD</b>       | Contrôleur IDE        |
| <b>Carte réseau</b> | Active                |

2. Il y a deux possibilités pour **installer les composants logiciels de Docker**. Nous le ferons **depuis les serveurs de dépôts de Debian Bookworm** ainsi que le **dépôt officiel de Docker** à l'aide de l'utilitaire « **apt** » :
  - Mettez à jour vos sites miroirs (dépôts des paquets Debian) dans le fichier « **sources.list** ».
  - Saisissez les commandes suivantes pour effectuer la mise à jour de votre système Debian : **#apt update && apt upgrade**
  - Installez les paquets suivants : **ca-certificates, curl, gnupg et lsb-release**

3. Créez un répertoire pour les clés de sécurité pour télécharger de façon sécurisée les composants logiciels nécessaires avec la commande suivante :
  - **#sudo install -m 0755 -d /etc/apt/keyrings**
  - Expliquez le rôle de la commande ainsi que celui des options associées.
4. Téléchargez et ajoutez la clé GPG officielle de Docker :
  - **#curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg**
  - Expliquez le rôle de la commande ainsi que celui des options associées.
5. Modifiez les droits d'accès du fichier « **docker.gpg** » en **ajoutant le droit en lecture à tout le monde**. Indiquez la commande que vous avez saisie.
6. A l'aide de la commande suivante, créez un fichier nommé « **docker.list** » à l'emplacement suivant « **/etc/apt/sources.list.d/** » :
  - **#echo « deb [arch=\$(dpkg --get-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \$(cat /etc/os-release | grep ID | cut -d= -f2) stable » | sudo tee /etc/apt/sources.list.d/docker.list**
7. Editez le fichier « **docker.list** ». Que remarquez-vous ? Expliquez ce résultat en déclinant le rôle des commandes et option associées.
8. Actualisez vos fichiers sources à l'aide de la commande : **#apt update**
9. Maintenant, installez Docker CE et ses composants logiciels depuis la commande :
  - **#apt install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin**
10. Vérifiez l'état du service « **Docker Engine** » :
  - **#systemctl status docker**
11. Configurez le démarrage automatique de Docker au lancement de « **SRVDOCKER** » :
  - **#systemctl enable docker.service**
  - **#systemctl enable containerd.service**
12. Arrêtez, démarrez et redémarrez le service Docker Engine à l'aide de la commande suivante :
  - **#systemctl start|stop|restart docker**
13. Affichez les informations relatives à la version de Docker installée sur votre serveur « **SRVDOCKER** », à l'aide des commandes suivantes :
  - **#docker version** ou **-v #docker info**                      **#docker compose version**

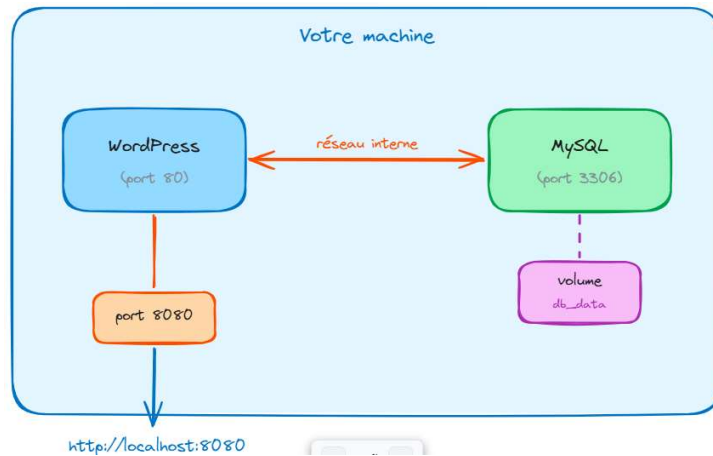
**Remarque :** En cas de message d'erreur, relancez votre serveur « **SRVDOCKER** ».

## **Etape 2 : Création de la structure du projet « WordPress » et définition de son fichier « docker-compose.yml »**

Dans un premier temps, vous allez déployer les services et applicatifs suivants au sein de différents conteneurs :

- **MariaDB ou MySQL** : système de gestion de BDD relationnelles permettant de stocker, organiser et gérer les données.
- **PhpMyAdmin** : application web offrant une interface graphique pour gérer une BDD relationnelle (souvent basée sous MariaDB ou MySQL).
- **WordPress** : est un système de gestion de contenu (CMS) pour créer et gérer un site web facilement sans pour autant l'avoir codé.

Ci-après un extrait de l'architecture de Docker attendue :



14. Depuis le serveur « **SRVDOCKER** », connectez-vous en tant qu'Administrateur et depuis son répertoire personnel, créez un répertoire nommé « **wordpress/** ». Indiquez les commandes saisies.
15. Déplacez-vous dans le répertoire « **wordpress/** » et créez un fichier texte nommé « **docker-compose.yml** » et saisissez-y le contenu suivant :

```

GNU nano 7.2
services:
  #Database
  bdd:
    image: mysql:latest
    volumes:
      - './db_data:/var/lib/mysql'
    ports:
      - "3306:3306"
    restart: always
    environment:
      MYSQL_DATABASE: wordpress
      MYSQL_USER: tepson
      MYSQL_PASSWORD: tepson2526
      MYSQL_ROOT_PASSWORD: admin2526

  #PhpMyAdmin
  phpmyadmin:
    depends_on:
      - bdd
    image: phpmyadmin/phpmyadmin:latest
    restart: always
    ports:
      - '8084:80'
    environment:
      PMA_HOST: bdd
      MYSQL_ROOT_PASSWORD: admin2526

  #Wordpress
  wordpress:
    depends_on:
      - bdd
    image: wordpress:latest
    ports:
      - '8080:80'
    environment:
      WORDPRESS_DB_HOST: bdd
      WORDPRESS_DB_USER: tepson
      WORDPRESS_DB_PASSWORD: tepson2526
      WORDPRESS_DB_NAME: wordpress
    restart: always

```

16. A partir du fichier « **docker-compose.yml** », indiquez le rôle des paramètres de configuration de chaque service applicatif à déployer dans un conteneur.
17. A l'aide de la commande, « **#docker compose up -d** », orchestrez l'ensemble du processus de création de l'architecture Docker précédente. Indiquez le rôle de l'option « **-d** ».
18. Montrez le résultat obtenu à partir d'une capture d'écran. Déduisez-en le nom du réseau virtuel qui permet la communication entre les différents conteneurs.
19. Vérifiez que les conteneurs tournent à l'aide de la commande « **#docker compose ps** ». Indiquez le résultat obtenu (capture d'écran).
20. Proposez une procédure pour tester, depuis votre poste de travail, l'accès au SGBDR MySQL depuis l'application web PHPMyAdmin.
21. Proposez une procédure pour tester, depuis votre poste de travail, l'accès à l'application Wordpress et finaliser son installation pour héberger le site du « **CHU de Pointe-à-Pitre** ». Montrez le résultat obtenu à l'aide d'une capture d'écran.

**C'est à vous de jouer !!!**

**Etape 3 : Création de la structure du projet « GLPI OCS » et définition de son fichier « docker-compose.yml »**

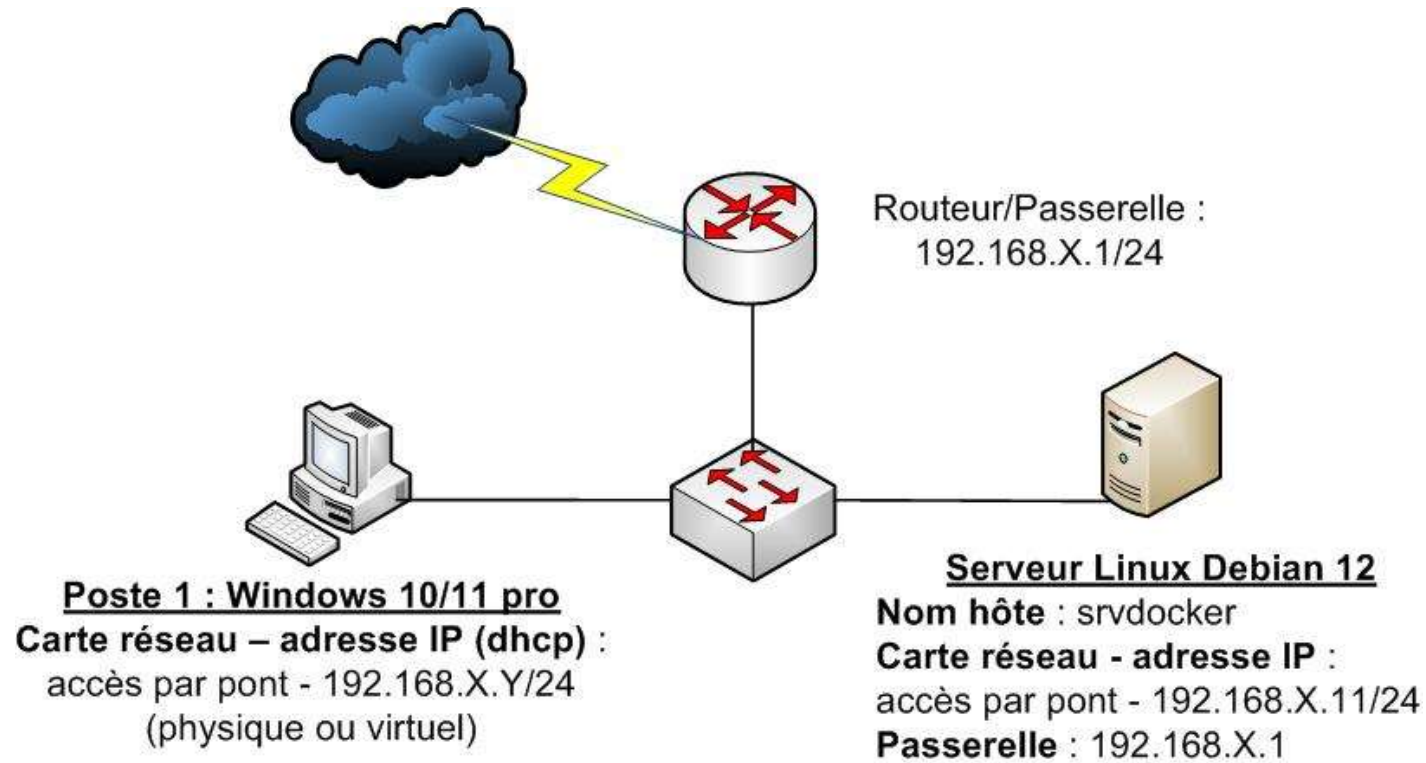
Maintenant, dans un deuxième temps, vous devez déployer les applications *PhpMyAdmin*, *GLPI* et *OCS Inventory* dans des conteneurs différents avec des bases de données gérées à partir du SGBDR *MySQL/MariaDB*, installé dans son propre conteneur.

22. Proposez un schéma qui présente l'architecture Docker attendue.
23. Indiquez la démarche à suivre pour orchestrer ces services dans des conteneurs avec Docker Compose.
24. Proposez un plan de tests ainsi que des captures d'écran mettant en exergue les résultats obtenus.

**Bon travail !!!**

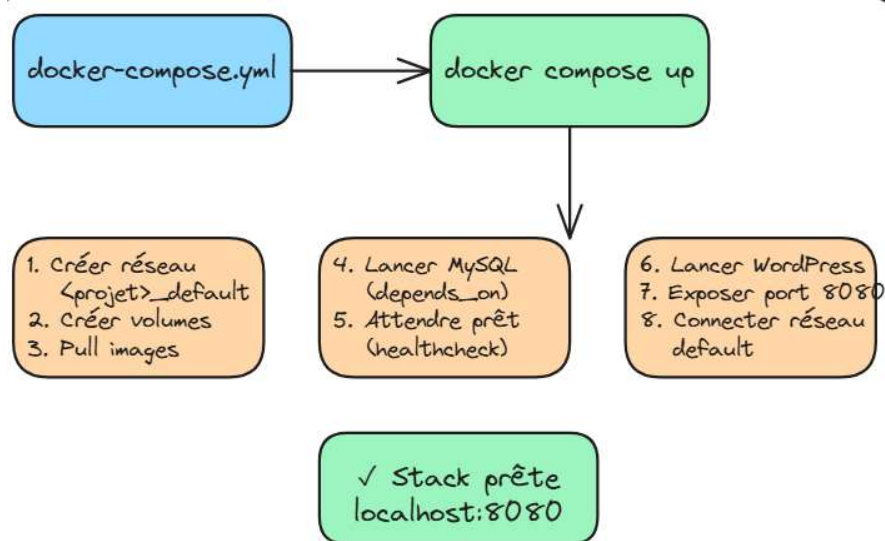
## ANNEXES

### Document 1 : Architecture réseau à réaliser



## Document 2 : Lancement d'une stack avec Docker Compose

Lorsque vous exécutez « **docker compose up** », Docker Compose orchestre l'ensemble du processus de création de l'infrastructure. Voici ce qui se passe en arrière-plan :



## Document 3 : Commandes essentielles de Docker Compose

Voici les commandes que vous utiliserez régulièrement. La plupart s'exécutent depuis le dossier contenant votre fichier « **docker-compose.yml** ». Si vous êtes ailleurs, utilisez l'option « **-f** » pour spécifier le chemin du fichier.

| Commande                                    | Description                                     |
|---------------------------------------------|-------------------------------------------------|
| <code>docker compose up -d</code>           | Démarre tous les services en arrière-plan       |
| <code>docker compose down</code>            | Arrête et supprime les conteneurs et réseaux    |
| <code>docker compose down -v</code>         | Idem + supprime les volumes (perte des données) |
| <code>docker compose ps</code>              | Liste les conteneurs du projet                  |
| <code>docker compose logs -f</code>         | Affiche les logs en temps réel                  |
| <code>docker compose exec mysql bash</code> | Ouvre un shell dans le conteneur MySQL          |
| <code>docker compose restart</code>         | Redémarre tous les services                     |
| <code>docker compose pull</code>            | Met à jour les images                           |
| <code>docker compose config</code>          | Valide et affiche la configuration finale       |