TP : DOCKER-COMPOSE

Partie 1 : Préparation de l'environnement

1. Assurez-vous que docker compose est déjà installé avec la commande : *docker-compose --version*
2. Afin de travailler dans un environnement vierge et ne impacter d'autres stacks docker-compose, créez un dossier et placez-vous à l'intérieur. Toutes les commandes qui suivront seront lancées depuis ce répertoire.
   a. Créez un répertoire
      mkdir mystack
   b. Dans ce répertoire, créez un fichier docker-compose. yml
      Nous utiliserons l'application gitea comme démonstrateur. gitea est un serveur Git écrit en Go concurrent de Gitlab ou Bitbucket par exemple. L'application se compose du serveur lui-même ainsi que d'une base de données. Ces deux parties seront déployées par Docker Compose. Copier/coller cette stack dans un fichier docker-compose.yml

```
version: '2'
services:
  # Service pour le serveur Gitea
  gitea_server:
    image: gitea/gitea:1.19  # Utilisation de l'image Gitea version 1.19
    ports:
      - "3000:3000"  # Redirection du port 3000 de l'hôte vers le port 3000 du conteneur (interface web de Gitea)
      - "10022:22"   # Redirection du port 10022 de l'hôte vers le port 22 du conteneur (SSH)
    volumes:
      - gitea_server_data:/data  # Montage d'un volume pour stocker les données du serveur Gitea
    networks:
      - gitea_network  # Utilisation du réseau "gitea_network" pour permettre la communication entre les services

  # Service pour la base de données MySQL utilisée par Gitea
  gitea_db:
    image: mysql:8  # Utilisation de l'image MySQL version 8
    volumes:
      - gitea_db_data:/var/lib/mysql  # Montage d'un volume pour stocker les données de la base de données MySQL
    environment:
      - MYSQL_ROOT_PASSWORD=password  # Définition du mot de passe root de MySQL
      - MYSQL_DATABASE=gitea  # Création de la base de données "gitea"
    networks:
      gitea_network:  # Utilisation du même réseau que le serveur Gitea
        aliases:
          - mysql  # Alias pour le service MySQL, utilisé pour la communication entre les services

volumes:
  gitea_server_data:  # Définition d'un volume nommé "gitea_server_data" pour le serveur Gitea
    driver: local  # Utilisation d'un volume local (stockage sur le système hôte)
  gitea_db_data:  # Définition d'un volume nommé "gitea_db_data" pour la base de données MySQL
    driver: local  # Utilisation d'un volume local (stockage sur le système hôte)
```

```
networks:
  gitea_network:  # Définition d'un réseau nommé "gitea_network" pour permettre la communication entre les services
```

c. Avant de lancer votre stack, essayez de retrouver :
   - La version de gitea utilisée
   - La façon dont MySQL récupère son master password
   - Le nom que va prendre MySQL grâce au service discovery de Docker
   - Le port SSH sur lequel votre serveur gitea écoutera

3. Vous pouvez maintenant lancer votre stack : *$ docker-compose up –d*
4. Vérifiez que votre stack est correctement lancée : *$ docker-compose ps*
5. Vérifiez que :
   a. Les volumes ont bien été créés : *docker volume ls*
   b. Trouvez leur emplacement sur votre système hôte : *docker volume inspect nom_du_volume*
   c. Le réseau a bien été créé : *docker network ls*
   d. Les conteneurs sont bien démarrés et visibles par le démon Docker : *docker ps*

Partie 2 : Configuration de gitea

1. Rendez-vous sur l'adresse de votre daemon Docker au port en écoute par gitea, probablement localhost:3000 et procédez à l'installation de gitea
2. Sur quelle adresse se trouve votre container MySQL ?
3. Quel est le password root de la base de données ?
4. Il n'est pas utile de remplir toutes les informations. Seule la partie présentée dans le screenshot a besoin d'être mise à jour en fonction de votre docker-compose.yml.
   - Type de base de données : MySQL
   - Hôte : gitea_db
   - Nom d'utilisateur : root
   - Mot de passe : password
   - Nom de la base de données : gitea

   Une fois l'installation terminée, vous devez vous créer un compte et ensuite vous loguer. Créer un repository et ajoutez-y un fichier quelconque. Nous souhaitons simplement tester la persistance des données.

Partie 3 : Vérification de la persistance

1. Notre application fonctionne et nous afin de vérifier que nos conteneurs ne disposent pas de données locales.
   a. Vous devez détruire vos conteneurs :
      $ docker-compose stop
      $ docker-compose rm –f
   b. Vérifiez que les volumes sont toujours présents sur le système : *docker volume ls*
   c. Vérifiez que vous avez bien perdu l'accès à l'application !!!
2. On peut maintenant relancer notre application : *docker-compose up -d*
3. Vérifiez que vous avez récupéré l'accès à l'application
4. Vérifiez que les données que vous y aviez mises y sont toujours.