Installation et utilisation de Docker

Pourquoi utiliser Docker :

Docker est une plateforme de virtualisation légère qui permet de conteneuriser des applications. Utiliser Docker pour le déploiement d'une application Symfony offre plusieurs avantages :

- **Portabilité et Consistance** : Docker garantit que votre application fonctionne de manière identique sur toutes les machines. Que ce soit en développement local, en intégration continue ou en production, Docker permet de réduire les risques liés aux différences d'environnement, ce qui simplifie le processus de déploiement.

- **Isolation des dépendances** : Docker permet de contenir toutes les dépendances nécessaires à votre application dans un conteneur (base de données, serveur web, PHP, etc.), ce qui évite des conflits avec d'autres services installés sur la machine hôte.

- **Facilité de mise en production** : Grâce à Docker, le processus de déploiement devient plus fluide. En utilisant des images Docker prêtes à l'emploi, vous pouvez déployer une application Symfony avec toutes ses dépendances et sa configuration en quelques minutes.

- **Gestion simplifiée des environnements** : Docker permet de gérer plusieurs environnements de développement, de test et de production sans avoir à installer et configurer des logiciels sur chaque machine. Vous pouvez reproduire exactement l'environnement de production sur votre machine de développement ou d'intégration continue.

Installation de docker :

- Tout d'abord, installez Docker sur votre ordinateur.

- **Création d'un dockerfile:** Le Dockerfile est un fichier de configuration qui décrit l'image Docker à utiliser. Pour un projet Symfony, voici un exemple de Dockerfile : Ce fichier crée une image Docker avec PHP, Nginx et les dépendances nécessaires à l'exécution de Symfony.

- **Configurer Docker Compose** Docker Compose permet de définir et de gérer plusieurs conteneurs Docker. Par exemple, pour Symfony, vous pouvez avoir un conteneur pour votre application et un autre pour la base de données. Voici un exemple de fichier.

nbclick_app > docker-compose.yml

```yaml
services:
  db:
    image: mysql:8.0
    container_name: symfony_db
    environment:
      MYSQL_DATABASE: symfony_db
      MYSQL_USER: symfony_user
      MYSQL_PASSWORD: password
      MYSQL_ROOT_PASSWORD: rootpassword
    ports:
      - "3306:3306"
    volumes:
      - db_data:/var/lib/mysql

  app:
    build:
      context: .
      dockerfile: docker/php/Dockerfile
    container_name: symfony_app
    volumes:
      - .:/var/www/html
    depends_on:
      - db
    entrypoint: [ "./entrypoint.sh" ]
    command: [ "php-fpm" ]
    logging:
      driver: "json-file"
      options:
        max-size: "10m"
        max-file: "3"
```

Docker dans notre projet :

Marche à suivre:

-Création d'un dossier à la racine du projet : docker

-Ce dossier comprend 2 sous dossier : **php et nginx**

php contient le dockerfile. (fichier de configuration pour l'image docker)

nginx contient le default.conf (configuration nginx, donc du serveur.)

-**Création docker-compose.yml** qui permet la création et le paramétrage des dockers (app / db / nginx)

-**Création d'un entrypoint.sh** (Il permet la création de la base de données ainsi que l'exécution des migrations au lancement du docker.)

dockerfile :

nbclick_app > docker > php > dockerfile

```
1   FROM php:8.2-fpm
2   RUN apt-get update && apt-get install -y \
3       git \
4       unzip \
5       libzip-dev \
6       default-mysql-client \
7   && docker-php-ext-install zip pdo pdo_mysql
8   COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
9   COPY entrypoint.sh /usr/local/bin/entrypoint.sh
10  RUN chmod +x /usr/local/bin/entrypoint.sh
11  ENTRYPOINT ["/usr/local/bin/entrypoint.sh"]
12  CMD ["php-fpm"]
13  WORKDIR /var/www/html
```

-**From** : tu recuperes l'image de PHP 8.2
-**Run** : recuperation toutes les mise a jour et installation des dependance desiré dans le docker.
-**COPY**: Copie de la derniere image de composer pour la mettre dans le docker
-**COPY** Copie le entrypoint. sh (contenant le script de database et de migrations)
- **Run** Exécution de l'entrypoint (porte d'entré) avec les droits administrateurs lorsqu'un docker est lancé

-**CMD** Commande par defaut a executer si aucune commande n’est donnée au lancement du conteneur.
-**WorkDIR:** repertoire dans lequel le repertoire est positionné.