**Docker Compose**

Docker Compose est conçu pour la gestion des applications multi-conteneurs sur une seule machine,

 tandis que Kubernetes est conçu pour la gestion de conteneurs sur plusieurs machines.

Avec Docker Compose, les développeurs peuvent décrire les services, réseaux et volumes

nécessaires pour exécuter une application multi-conteneurs sur une seule machine.

Docker Compose crée et gère automatiquement les conteneurs Docker correspondants,

mais sur une seule machine. Docker Compose est donc particulièrement utile pour

les environnements de développement et de test.

En revanche, Kubernetes est une plateforme d'orchestration de conteneurs qui permet de déployer,

 gérer et mettre à l'échelle des conteneurs sur un cluster de machines.

Kubernetes permet de déployer des applications dans des environnements distribués et hautement disponibles, et peut gérer des milliers de conteneurs sur plusieurs machines.

En résumé, Docker Compose est destiné à la gestion d'applications multi-conteneurs sur une seule machine, tandis que Kubernetes est destiné à la gestion de conteneurs sur un cluster de machines.

En Docker Compose, un ensemble de conteneurs interconnectés est appelé un "projet".

Un projet est défini dans un fichier de configuration Docker Compose, qui spécifie les services nécessaires pour exécuter l'application.

Pour exécuter une application multiconteneur en utilisant Docker Compose

On doit créer :

1-Un fichier **Dockerfile** : pour construire une image Docker en définissant les instructions pour la construction de l’image.

2-Un fichier **.yaml**, tel qu’un fichier docker-compose.yml pour définir et exécuter plusieurs conteneurs Docker à l’aide d’une seule commande. Il permet de configurer différents types de services et de lancer tous les conteneurs avec une seule commande. Ces deux fichiers peuvent être utilisés ensemble pour créer et exécuter des applications Docker.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Dockerfile\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FROM php:7.4-apache

RUN docker-php-ext-install mysqli

RUN docker-php-ext-install pdo pdo\_mysql

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* docker-compose.yaml\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

version: '3.1'

services:

  php:

    build:

      context: .

      dockerfile : Dockerfile

    ports:

        - '80:80'

    volumes:

        - ./src:/var/www/html/

  db:

    image: mysql

    # NOTE: use of "mysql\_native\_password" is not recommended: https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-sha2-password

    # (this is just an example, not intended to be a production configuration)

    command: --default-authentication-plugin=mysql\_native\_password

    restart: always

    environment:

      MYSQL\_ROOT\_PASSWORD: example

  adminer:

    image: adminer

    restart: always

    ports:

      - 8080:8080

-Pour lancer l'application : se positionner dans le dossier racine de l’application et exécuter la commande : **docker compose up**

**En Résumé :**

Après avoir exploré le concept d'applications cloud exécutées dans des conteneurs en utilisant Docker localement, il est important de noter qu'Azure propose des services dans ce contexte :

•**Azure Container Registry (ACR)**: agit comme un équivalent de Docker Hub pour stocker vos images Docker.

•**Azure Container Instances (ACI)** : permet de déployer et d'exécuter vos images dans des conteneurs de manière isolée et sans gestion d'infrastructure.

•**Azure Kubernetes Service (AKS)** : est utile lorsque votre application utilise plusieurs conteneurs qui doivent s'exécuter sur plusieurs serveurs.

Alors que Docker Compose facilite l'exécution d'une image utilisant plusieurs conteneurs sur une même machine locale, AKS fournit une solution pour le déploiement et la gestion d'applications basées sur plusieurs conteneurs s'exécutant sur des serveurs distincts.