Task 1: Docker

1)- Create a Dockerfile for a web application that includes all necessary dependencies and configuration.

Créez un Dockerfile pour une application web incluant toutes les dépendances et configurations nécessaires.

Build a Docker image based on the Dockerfile created in the previous task.

Construisez une image Docker basée sur le Dockerfile créé dans la tâche précédente.

Run the Docker container based on the image, exposing the necessary ports.

Exécutez le conteneur Docker basé sur l'image, en exposant les ports nécessaires.

Task 2: Docker Compose

Create a Docker Compose YAML file that defines a multi-container application consisting of a web server and a database.

Créez un fichier Docker Compose YAML qui définit une application multi-conteneur composée d'un serveur web et d'une base de données.

Use Docker Compose to bring up the entire application stack and ensure proper communication between the containers.

Utilisez Docker Compose pour démarrer l'ensemble de la pile d'application et assurez-vous d'une communication appropriée entre les conteneurs.

Verify that the application is running correctly by accessing it through a web browser.

Vérifiez que l'application fonctionne correctement en y accédant via un navigateur web.

Task 3: Docker Swarm and Kubernetes (K8s)

Initialize a Docker Swarm cluster with multiple nodes.

Initialisez un cluster Docker Swarm avec plusieurs nœuds.

Deploy a service on the Docker Swarm cluster and ensure it is replicated across multiple nodes.

Déployez un service sur le cluster Docker Swarm et assurez-vous qu'il est répliqué sur plusieurs nœuds.

Migrate the same application from Docker Swarm to a Kubernetes (K8s) cluster.

Migrez la même application de Docker Swarm vers un cluster Kubernetes (K8s).

reponse

```
docker container run nginx
docker container run --name machine -ti ubuntu
docker container ps -a
docker container exec -it machine /bin/bash
docker start machine
docker container stop $(docker container ps -aq)
docker container start $(docker container ps -aq)
docker container rm $(docker ps -aq)


etape de creation site web mdify page index:
docker container run -d -p 8080:80 --name web1 nginx

docker container exec -it web1 /bin/bash

root@ca6d15be623f:/# echo "salam" > /usr/share/nginx/html/index.html



---------

Stockage non persistant ==

Stockage persistant == volume et bind mounts

docker volume create VOL1
docker volume ls

docker container run -p 8088:80 -d --name web3 -v
VOL1:/usr/share/nginx/html nginx

docker volume inspect VOL1

echo "RSSP" > /var/lib/docker/volumes/VOL1/_data/index.html
```

----

echo "RSSP --- DKK" > index.html

-------- dockerfile

```
FROM ubuntu

RUN apt update -y && apt install nginx -y

WORKDIR /var/www/html

COPY index.html .

EXPOSE 80/tcp

CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
```

----

docker image build -t machine .

docker run -d -p 8077:80 --name IMAGE machine

----

docker compose:

docker-compose up -d

fichier: docker-compose.yml

```
version: '3.3'

services:

  wordpress:

    image: wordpress

    depends_on:

      - mysql

    ports:
            - "8098:80"

        environment:

            WORDPRESS_DB_HOST: mysql

            WORDPRESS_DB_NAME: wordpress

            WORDPRESS_DB_USER: wordpress

            WORDPRESS_DB_PASSWORD: wordpress

        volumes:

            - ./wordpress-data:/var/www/html

        networks:

            - my_net


    mysql:

        image: mariadb

        environment:

            MYSQL_ROOT_PASSWORD: wordpress

            MYSQL_DATABASE: wordpress

            MYSQL_USER: wordpress

            MYSQL_PASSWORD: wordpress

        volumes:

            - mysql-data:/var/lib/mysql

        networks:

            - my_net


volumes:

    mysql-data:


networks:

    my_net:
```

----

swarm

docker swarm init --advertise-addr 192.168.0.8

```
docker node ls
docker network ls

docker service create --replicas 5 -p 8099:80 --name web2 nginx

  docker service ls

docker service ps web2
```

Docker stack:

```
version: '3.3'


services:

  wordpress:

    image: wordpress

    depends_on:

      - mysql

    ports:

      - "8098:80"

    deploy:

      replicas: 2

      placement:

        constraints:

          - node.role == manager

    environment:

      WORDPRESS_DB_HOST: mysql

      WORDPRESS_DB_NAME: wordpress

      WORDPRESS_DB_USER: wordpress

      WORDPRESS_DB_PASSWORD: wordpress

    volumes:

      - ./wordpress-data:/var/www/html

    networks:

      - my_net


  mysql:

    image: mariadb

    environment:
      MYSQL_ROOT_PASSWORD: wordpress

      MYSQL_DATABASE: wordpress

      MYSQL_USER: wordpress

      MYSQL_PASSWORD: wordpress

    volumes:

      - mysql-data:/var/lib/mysql

    networks:

      - my_net


volumes:

  mysql-data:


networks:

  my_net:
```

```
docker stack deploy -c compose.yml LL
```