

Enoncé du Mini-Projet

Considérons une application web 3-tiers composée de frontend (Angular), backend (Java/Spring) et une base de données (MySQL).

Le code source de cette application est disponible sur ces repos : [frontend](#) et [backend](#).

Partie 1 :

Dans cette partie, il est demandé de réaliser les manipulations suivantes en utilisant un outil de conteneurisation comme Docker.

1. Créer un conteneur pour la base de données MySQL en instanciant l'image officielle *mysql*.

NB : Choisir l'option de stockage la plus adéquate, puis effectuer les actions nécessaires pour rendre le stockage de cette base de données persistant. Justifier votre choix.

2. Créer un réseau Docker (tout en lui précisant un driver convenable). Puis connecter le conteneur de la base de données mysql avec ce réseau.
3. Créer un fichier Dockerfile pour le projet Backend (essayez d'appliquer le maximum de bonnes pratiques).
4. Lister et expliquer "en détails" ces bonnes pratiques appliquées.
5. Taper les commandes docker nécessaires pour :
 - a. Créer une image docker en local à partir de ce fichier Dockerfile.
 - b. Scanner l'image des vulnérabilités qu'elle peut contenir. En cas d'existence de ces vulnérabilités, effectuer les actions nécessaires pour les éviter.
 - c. Publier cette image dans Docker Hub.
 - d. Instancier cette image en créant un conteneur s'exécutant en local et partageant le même réseau avec le conteneur de la base de données.
 - e. Inspecter ce conteneur du backend.
 - f. Afficher les logs liés à ce conteneur du backend.

NB : Commenter les attributs utilisés dans chacune de ces commandes.

6. Refaire le travail demandé dans les questions 3), 4) et 5) pour le projet Frontend.
7. S'assurer que l'application a été bien conteneurisée et qu'elle fonctionne correctement. (Prendre des prises d'écran du navigateur)

8. Supprimer les 3 conteneurs en exécution.
9. Redéployer l'application (frontend, backend et base de données) en utilisant docker-compose.
 - a. Créer le fichier docker-compose.
 - b. Taper la commande docker permettant de l'exécuter.
 - c. S'assurer que l'application fonctionne correctement (prises d'écran du navigateur)
10. Supprimer les conteneurs qui s'exécutent ainsi que les images se trouvant dans le docker host.
11. Créer un registre privé d'images docker en local qui vous permettra de :
 - a. Stocker les images que vous avez construites.
 - b. Visualiser via une UI les images qu'il contient.
12. Redéployer l'application (frontend, backend et base de données) en utilisant docker-compose tout en récupérant l'image des conteneurs depuis ce nouveau registre privé.
NB : S'assurer que l'application fonctionne correctement (prises d'écran)

Partie 2 :

Dans cette partie, il est permis d'utiliser n'importe quel outil permettant de créer un cluster Kubernetes en local (ex : Minikube, MicroK8s, kind, k3d, k3s, etc.)

1. Créer les Kubernetes manifests (fichiers YAML) nécessaires pour déployer cette application dans un cluster Kubernetes (sous un namespace nommé "exam").
 - a. Deployment
 - b. Service
 - c. ConfigMap
 - d. Secret
 - e. Statefulset

NB1 : Vous avez le choix d'utiliser n'importe quel registre d'images (docker hub, harbor, registre privée en local, etc.) pour récupérer les images nécessaires.

NB2 : Créer 2 réplicas pour chaque partie de l'application (frontend, backend et base de données)

2. Définir des quotas pour la consommation des ressources Mémoire et CPU de chaque Pod exécuté dans le namespace « exam ».
3. Contrôler l'accès aux ressources existantes dans le namespace « exam » par un rôle RBAC.

4. Spécifier un budget de perturbation (Disruption Budget) pour votre application.
5. Créer et configurer pour cette application un :
 - a. Liveness Prob
 - b. Readiness Prob
 - c. Startup Prob

NB : Vérifier les événements générés par les différents pods de l'application.

6. Créer un Ingress pour accéder à l'application depuis un navigateur moyennant un nom de domaine.
7. S'assurer que l'application a été bien conteneurisée et qu'elle fonctionne correctement. (Prendre des prises d'écran du navigateur)
8. Repenser à créer et gérer la partie base de données de l'application en utilisant le concept de : Operator. (Créer un cluster de bases de données)

Bonus 1 : Utiliser l'outil Skaffold pour détecter les changements de source pendant le développement, construire et créer automatiquement les images Docker et puis pour pousser et déployer les artefacts dans le cluster Kubernetes.

Bonus 2 : Créer "from scratch" un/des Helm chart(s) pour cette application. Publier votre chart dans un repo public, puis déployer l'application via Helm (sous un autre namespace nommé "*Helm*").

Livrable :

Un **document** au format PDF contenant les réponses de chacune des questions ci-dessus (bien alimenté avec des explications détaillées + des prises d'écran + URL de votre repo GitHub pour accéder à votre code source).

Consignes :

- Ce travail est à réaliser **individuellement**.
- Le dernier délai pour envoyer les livrables du projet est le **dimanche 3 décembre** (avant minuit).

Bon courage

الله ولي التوفيق