Examen – TP

Exercice 1 : Commandes docker (4 points – durée 25 minutes)

Donner les commandes docker qui vont permettre de :

1.1 Créer un conteneur apache « apache-ctn » qui va se baser sur l'image httpd en « mode détacher » et qui va mapper le port 80 du conteneur sur le port 9191 de votre hôte.

1.2 Lister les conteneurs actifs

1.3 A l'aide d'une commande, modifier le fichier index.html par défaut de Apache (si vous ne connaissez pas le répertoire par défaut, regarder la documentation de l'image) pour modifier son contenu par « Coucou depuis mon conteneur Apache »

1.4 Créer une nouvelle image « coucou-apache » en prenant comme base votre conteneur «apache-ctn » qui a été modifié et en précisant le message de commit « Modification index.html ».

1.5 Stopper et supprimer le conteneur « apache-ctn » qui a été crée dans l'exercice 1.1

1.6 Recréer un conteneur « apache-new » mais qui va mapper sur le port 9999 et en prenant comme base l'image « coucou-apache » qui a été crée au point 1.4

1.7 Stopper et supprimer le conteneur « apache-new » et supprimer l'image « coucou-apache »

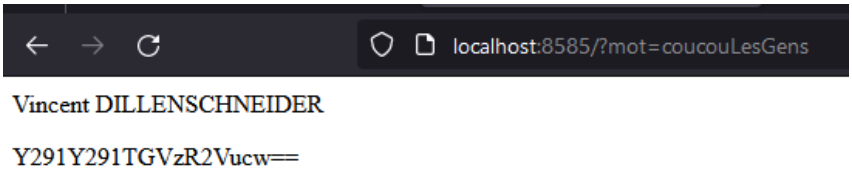
Exercice 2 : Dockerfile et docker-compose (4 points – durée 30 minutes)

2.1 En respectant les bonnes pratiques sur la rédaction d'un Dockerfile, créer un Dockerfile qui :
- va prendre comme image de base debian
- installer apache et PHP
- supprimer le fichier index.html
- créer un fichier index.php qui va :
* lire un paramètre de type GET qui se nomme « mot » (utiliser l'instruction php $mot=$_GET['mot'] ; pour récupérer le contenu du paramètre)
* encoder le contenu de ce paramètre en base64 dans une variable (utiliser la fonction base64_encode de php sur la variable précédente)
* afficher dans la page Web retournée par le script index.php :
  - votre prénom et votre nom sur une ligne
  - le contenu de cette variable encodée sur la deuxième ligne

2.2 Donner la ligne de commande Docker pour construire l'image « encoding-word » en utilisant le Dockerfile du point 2.1

2.3 Donner la commande docker pour lancer un conteneur « ctn-encoding » qui va utiliser l'image du point 2.2

2.4 Faire une capture d'écran de votre résultat (cf : ci-dessous).



2.5 Créer un fichier docker-compose pour utiliser votre image « encoding-word » et réaliser un mapping de port avec le port 8080, ce conteneur devra utiliser un réseau « mynet » de type bridge qui va définir un réseau sur 172.16.1.0/24. L'adresse IP de ce conteneur sera attribué manuellement sur 172.16.1.20

2.6 Donner la commande pour lancer votre docker-compose en mode détacher

2.7 Donner la commande pour terminer vos conteneurs (et les supprimer) avec docker-compose

Exercice 3 : Manipulation Kubernetes (5 points – durée 20 minutes)

Rédiger des fichiers YAML qui vont permettre de :

3.1 Créer un pod my-caddy qui va utiliser l'image caddy (il s'agit d'un logiciel de serveur Web tout comme Apache ou nginx) de Dockerhub.

3.2 Créer un service de type nodePort sur le port 30200 pour exposer le Pod précédent à l'extérieur du cluster

3.3 Tester depuis votre navigateur l'accès vers ce pod et réaliser une capture d'écran de la page d'accueil.

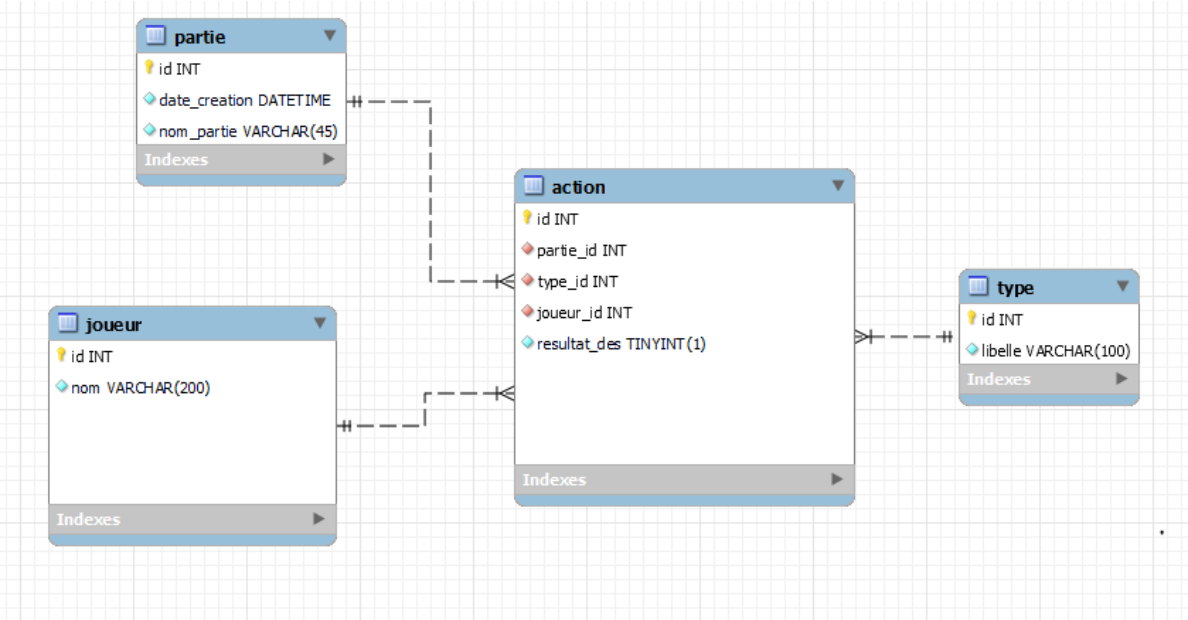
3.4 Supprimer ce pod et donner la commande qui a été réalisée

3.5 Réaliser un déploiement « mes-caddies » qui va mettre en place 3 replicas de pod caddy (conserver les mêmes labels pour que le service continue de fonctionner).

Exercice 4 : Projet iLoveCraft (7 points – durée 1h30)

Une société de diffusion de jeu de société (le client) vous demande de créer un POC qui concerne un jeu qui s'inspire de l'univers de LoveCraft. Traditionnellement sur ce genre de jeu, dans le cadre d'une partie, des joueurs doivent réaliser des actions (test de force, agilité, etc.) en effectuant des lancés de dés.
Un dé peut avoir comme résultat : échec, indice ou succès.

Lors de votre brainstorming avec votre équipe, vous avez créer une première version de base de données dont voici le schéma :



Vous avez également décidé ensemble des valeurs informatiques suivantes pour les lancés de dés :
0 = échec, l'action a échouée
1 = indice, l'action peut devenir un succès si un joueur dépense un indice (les indices sont des jeton que peut récupérer les joueurs au cours de la partie)
2 = succès, l'action a réussi.

Au sein de votre équipe, vous avez été désigné pour réaliser la gestion du lancé de dés d'une partie en réalisant les points suivants du projet :

1/ réaliser une API qui va effectuer un lancement de dé aléatoire et stocker le résultat dans la base de données. (3,5 points)

L'appel de cet API se fera via une requête GET sur l'URL suivante :
http://localhost:8080/ilovecraft/api/lancerDe.php?partieId=<identifiantPartie>&joueurId=<identifiantJoueur>&typeId=<IdentifiantType>

exemple avec des valeurs :
http://localhost:8080/ilovecraft/api/lancerDe.php?partieId=1&joueurId=1&typeId=1

L'appel de cet URL va :

- effectuer un lancement de dé aléatoire
- enregistrer le résultat dans la base de données
- retourner le résultat du dé (0, 1 ou 2) à l'utilisateur dans un format JSON de ce type :

```
{
   "result": 1
}
```

Contraintes techniques :

La solution d'API utilisera le socle technique suivant (choix de votre équipe) : apache / PHP peu importe l'OS.
Le système de base de données à utiliser : MySQL.

Pour partager facilement votre travail, on vous demande de réaliser ce projet avec l'aide de Docker en effectuant :
- un dockerFile qui va construire l'image de la base de données et intégrer son script d'initialisation (fournit en PJ) pour disposer d'une base de données fonctionnelle avec des données initiales.
- un dockerFile qui va construire l'image de l'application qui fournira le système d'API.
- un fichier docker-compose qui va permettre de lancer l'ensemble de votre application (API + base de données) de manière fonctionnelle et cohérente.

****Bonus** (2 points) :**

Pour surprendre votre client dans le cadre du POC, vous pouvez également créer une nouvelle application frontend (donc une nouvelle image Docker) qui va proposer une interface pour lancer l'appel de votre API à l'aide d'un bouton pour ensuite afficher le résultat du dé à l'écran.
Intégrer ce frontend à votre docker-compose.

L'URL pour utiliser ce frontend sera :
http://localhost:8090/ilovecraft

2/ Déployer vers le cloud (3,5 points)

Le client souhaite pouvoir tester sa solution dans le cloud car son jeu sera disponible online et il souhaite également pouvoir adapter l'application en fonction de la charge demandée par ses utilisateurs.

C'est pourquoi il vous demande de porter votre solution Docker (cf votre solution du point 1/) sur la plateforme Kubernetes.

Vous devez donc reprendre / publier vos images précédentes et définir une architecture K8s en utilisant selon vous toutes les ressources nécessaires (déploiement, pods, services, configmaps, secrets, etc...).

En fonction du temps que vous disposez, vous pouvez au choix :
- illustrer votre solution avec un schéma et des explications
- et / ou créer les fichiers manifestes YAML de votre architecture.