Projet de Groupe : Conteneurisation des services du Projet Annuel

Objectifs

> ***Utiliser Docker pour containeriser les composants de votre projet annuel.***

- Comprendre les principes de la conteneurisation des services d'une application.
- Utiliser Docker pour containeriser les composants d'une application web.
- Orchestrer les services avec Docker Compose pour faciliter le déploiement.
- Documenter l'architecture de l'application et les instructions d'utilisation.
- Utiliser des images Docker existantes pour enrichir l'outillage de développement et de gestion.

Format du Projet

- Groupe : Le même groupe que pour le projet annuel.
- Durée : 1 mois.

Description détaillée et étapes du projet

Schématisation de l'orchestration des services

1. **Création du Diagramme d'Architecture** : Utilisez un outil comme draw.io pour concevoir un diagramme qui illustre comment les différents services de votre application sont conteneurisés et comment ils interagissent entre eux.
    - Le diagramme doit inclure au **minimum trois services, dont une base de données**.
    - Illustrez clairement les liaisons réseau, les volumes partagés, et les dépendances entre les services.
2. **Détail des Services** : Pour chaque service représenté dans le diagramme, fournissez une brève description incluant son rôle dans l'application et les technologies utilisées (par exemple, Node.js pour le backend, React pour le frontend, PostgreSQL pour la base de données, etc.).
3. **Justification des Choix** : Expliquez pourquoi vous avez choisi cette architecture spécifique. Incluez les avantages de l'orchestration avec Docker et comment cela profite à votre application.

Construction d'Images Docker

1. **Définition des Images Docker** : Pour chaque composant de votre application, définissez une image Docker qui encapsule l'environnement et les dépendances nécessaires. Utilisez des Dockerfiles pour automatiser la création de ces images.
2. **Builds Multi-étapes** :
    - Implémentez des builds multi-étapes dans vos Dockerfiles pour séparer les environnements de développement et de production.
    - Assurez-vous que les images de production ne contiennent que les éléments nécessaires au moment de l'exécution de l'application, excluant les outils de développement et les fichiers temporaires.
3. **Optimisation des Images** :
    - Minimisez la taille des images en combinant des commandes, en utilisant des images de base minimalistes, et en évitant l'inclusion de fichiers inutiles.
    - Documentez chaque étape dans vos Dockerfiles pour expliquer son but et son impact sur l'image finale.
4. **Validation des Images** :
    - Testez les images en local pour vous assurer qu'elles fonctionnent comme prévu.
    - Vérifiez que les conteneurs démarrent correctement et que l'application s'exécute sans erreurs.

Utilisation d'un Registre d'Images

1. **Choix d'un Registre d'Images** :
   - Vos images Docker doivent être stockées dans un registre pour faciliter le partage et le déploiement.
   - Optez pour un registre d'images Docker privé (**pas obligatoire, bonus**) pour stocker vos images. Scaleway Container Registry est recommandé en raison de sa facilité d'utilisation et de son coût abordable, mais vous êtes libre de choisir un autre service si vous le préférez.
2. **Pousser les Images sur le Registre** :
   - Configurez vos identifiants de registre Docker localement pour sécuriser la communication.
   - Taguez vos images Docker construites avec le nom du registre, le nom du dépôt, et une étiquette (tag) versionnée.
   - Utilisez la commande `docker push` pour envoyer vos images vers le registre.
3. **Récupération et Utilisation des Images** :
   - Apprenez à utiliser la commande `docker pull` pour récupérer les images du registre quand nécessaire.
   - Assurez-vous que vos fichiers Docker Compose référencent les images du registre privé correctement.

Orchestration des Services avec Docker Compose

1. **Fichiers Docker Compose** :
   - Créez deux fichiers Docker Compose : `docker-compose.dev.yml` pour le développement et `docker-compose.prod.yml` pour la production.
   - Assurez-vous que chaque fichier configure correctement les services, les réseaux, les volumes, et les variables d'environnement appropriées à chaque contexte.
2. **Configuration pour le Développement** :
   - Dans `docker-compose.dev.yml`, configurez vos services pour faciliter le développement, comme le rechargement à chaud, l'accès aux logs simplifié...
   - Utilisez des volumes pour monter le code source dans les conteneurs, permettant une mise à jour en temps réel du code sans nécessité de reconstruire l'image.
3. **Configuration pour la Production** :
   - Le fichier `docker-compose.prod.yml` doit optimiser pour la sécurité et la performance. Minimisez l'exposition des ports, utilisez des variables d'environnement pour les configurations sensibles, et assurez-vous que les services redémarrent automatiquement en cas d'erreur.
   - Configurez les volumes de manière à persister les données importantes et à séparer clairement les données de l'application des données utilisateur.
4. **Lancement et Gestion des Services** :
   - Apprenez à démarrer, arrêter, et surveiller les services avec Docker Compose en utilisant les commandes appropriées pour chaque environnement (`docker compose -f docker-compose.dev.yml up` , par exemple).
   - Utilisez les logs pour le débogage et assurez-vous de savoir comment accéder aux logs de chaque service.

Intégration de Services de Gestion Tiers (Bonus)

1. **Choix des Outils de Gestion** :
    - Intégrez des outils de gestion tiers pour améliorer la surveillance, la gestion et le débogage de vos conteneurs Docker. Voici des exemples, **mais vous êtes libre de choisir d'autres outils** :
        - Portainer pour une gestion visuelle des conteneurs
        - Prometheus et Cadvisor pour la surveillance des performances, et Grafana pour la visualisation des données.
2. **Configuration de Portainer** :
    - Utilisez Portainer comme exemple d'outil de gestion. Intégrez-le dans votre environnement Docker pour permettre une gestion facile des conteneurs, images, réseaux et volumes.
3. **Surveillance avec Prometheus et Cadvisor** :
    - Configurez Prometheus et Cadvisor pour collecter des métriques sur les performances des conteneurs et de l'infrastructure.
4. **Visualisation avec Grafana** :
    - Intégrez Grafana pour visualiser les données de surveillance collectées par Prometheus.
    - Aidez-vous d'un Dashboard déjà fait pour y parvenir.
      Exemple :https://grafana.com/grafana/dashboards/893-main/

Rendu et Soutenance

1. **Préparation du Rendu** :
    - Vous devrez rendre tout sur MyGES avant la date limite.
    - Compilez tous les livrables du projet, y compris le code source, les fichiers Docker et Docker Compose, les diagrammes d'architecture, et la documentation.
    - Organisez les livrables de manière claire et intuitive, en veillant à ce que tout soit facilement accessible et compréhensible.
2. **Présentation de la Soutenance** :
    - Préparez une présentation de **10 minutes** pour exposer les points clés de votre projet. Cette présentation devrait inclure une introduction très rapide à l'application, une explication de l'architecture Dockerisée, et une démonstration des services de gestion tiers intégrés.
    - Mettez en évidence les défis rencontrés et comment vous les avez surmontés, les choix techniques effectués, et les avantages apportés par la conteneurisation.
3. **Démonstration Technique** :
    - Prévoyez une courte démonstration technique pour montrer le fonctionnement de votre application et l'efficacité de l'orchestration Docker.
    - Illustrez comment utiliser la documentation pour configurer et déployer l'application, et comment les outils de gestion tiers facilitent la surveillance et la maintenance.
4. **Session de Questions-Réponses** :
    - Soyez prêts à répondre aux questions concernant votre projet. Cela peut inclure des questions sur vos choix techniques, les problèmes rencontrés, et les aspects spécifiques de Docker et des services de gestion que vous avez utilisés.