Création d'une Infrastructure Docker et Orchestration Avancée

ESGI
école supérieure de génie informatique

Contents

Mise en Place d'un Pipeline CI/CD pour une Application E-Commerce Microservices

---

1. Objectifs du Projet

L'objectif de ce projet est de vous familiariser avec les meilleures pratiques de conteneurisation en utilisant Docker, Docker Compose et, en option, Docker Swarm. Vous serez amenés à créer des images Docker optimisées, à configurer des environnements de développement et de production multi-stage, et à déployer une application e-commerce basée sur une architecture microservices.

Vous travaillerez en groupe pour concevoir, tester et déployer une solution complète à partir du code source fourni. L'accent sera mis sur l'optimisation des images Docker, la sécurité, et la configuration adaptée aux environnements de développement et de production.

Groupe de 3 étudiants.

---

2. Prérequis

- Maîtrise des concepts de base de Docker.
- Connaissances en création de Dockerfile et utilisation de Docker Compose.
- Notions sur l'orchestration de conteneurs (Docker Swarm est un bonus).
- Compétences en Linux et gestion d'environnements.
- Compréhension des réseaux Docker.
- Les machines utilisées dans ce projet sont sous **Debian 12**.

---

3. Description du Projet

Vous recevrez un dépôt Git contenant le code source des différents composants d'une application e-commerce basée sur une architecture microservices. Votre mission sera de :

- Créer les **Dockerfile** pour chaque service avec les bonnes pratiques, y compris le multi-stage et la distinction entre développement et production.
- Configurer des fichiers **docker-compose.yml** adaptés aux environnements de développement et de production.

- Utiliser Docker Swarm pour l'orchestration en production.
- Assurer la sécurité et l'optimisation des images Docker.
- Documenter votre travail pour faciliter la compréhension et le déploiement par une tierce personne.

---

4. Tâches principales

A. Analyse du Projet

- Cloner le repository et explorer la structure du projet.
- Comprendre le rôle de chaque composant :
  - **Frontend Vue.js** : Interface utilisateur.
  - **Backend (3 microservices)** :
    - Service Produits (Node.js avec MongoDB).
    - Service Utilisateurs (Node.js avec JWT).
    - Service Panier (Node.js avec gestion des commandes ET du panier).

2. **Mise en place des branches Git** si utilisé
   - Implémenter le workflow Git basé sur GitFlow :
     - `main` : Branche pour la production.
     - `develop` : Branche pour le développement.
     - Branches éphémères :
       - `feature/<nom>` : Pour les nouvelles fonctionnalités.
       - `release/<version>` : Pour les versions prêtes à être déployées.
       - `hotfix/<nom>` : Pour corriger des bugs critiques.

B. Création des Dockerfile

1. **Dockerisation des microservices** :
   - Créer un **Dockerfile** pour chaque microservice.
   - Utiliser le multi-stage build pour optimiser la taille des images et le multi-environnement.
   - Assurer la distinction entre les environnements de développement et de production.
   - Intégrer les variables d'environnement nécessaires.
   - Appliquer les bonnes pratiques de sécurité (exécution avec un utilisateur non-root, gestion des secrets, etc.).

2. **Dockerisation du frontend** :
    - Créer un **Dockerfile** multi-stage pour le frontend Vue.js.
    - Utiliser un serveur web léger (comme Nginx) pour servir l'application en production.
3. **Utilisation de notre registry privée** :
    - Utilisez une registry privée pour la gestion des images.

C. Configuration de Docker Compose

1. **Environnement de développement** :
    - Créer un fichier **docker-compose.yml** pour orchestrer les services en développement.
    - Configurer les volumes pour le hot-reload.
    - Mettre en place un réseau Docker bridge pour la communication entre les services.
    - Inclure les instances MongoDB pour chaque service.
2. **Environnement de production** :
    - Créer un fichier **docker-compose.prod.yml** pour la production.
    - Optimiser les configurations pour un déploiement en production.
    - Configurer les variables d'environnement et les secrets de manière sécurisée.
    - Préparer la configuration pour Docker Swarm.

- **Docker Swarm** :
    - Déployer l'application en utilisant Docker Swarm pour l'orchestration (en bonus), sinon utiliser pour docker-compose.
    - Configurer les services pour une haute disponibilité.

D. Application des Bonnes Pratiques

- **Optimisation des images Docker** :
    - Réduire la taille des images.
    - Supprimer les fichiers inutiles après l'installation des dépendances.
- **Sécurité** :
    - Ne pas exécuter les conteneurs en tant que root.
    - Gérer les secrets et variables sensibles de manière sécurisée.
- **Logs et monitoring** :
    - Configurer les logs pour faciliter le débogage et la surveillance.

E. Tests et Validation

- **Tests fonctionnels** :
  - Vérifier le bon fonctionnement de chaque service.
  - Tester les interactions entre les services.
- **Scans de sécurité** :
  - Utiliser des outils comme **Trivy** pour scanner les images Docker.
- **Validation de l'environnement de production** :
  - S'assurer que l'application fonctionne correctement en production.
  - Déployer avec Docker Swarm et vérifier l'orchestration.

---

5. Bonus

- **CI/CD simplifiée** :
  - Mettre en place une pipeline de build automatique des images Docker.
- **Outils de sécurité supplémentaires** :
  - Intégrer d'autres outils de sécurité pour renforcer la protection de l'application.
- **Monitoring avancé** :
  - Mettre en place des outils de monitoring comme Prometheus et Grafana.
- **Mise en prod avec Docker Swarm** :
  - Utiliser le compose.prod pour docker swarm pour un déploiement production.
- Bonus Libre :
  - Tout bonus de votre part sera considéré.

---

6. Livrables attendus

1. **Dépôt Git** (optionnel pour les SRC mais souhaitable) :
   - Contenant les **Dockerfile**, les fichiers **docker-compose**, et le code source.

- Une arborescence claire et bien organisée.

2. **Documentation** :
   - Un fichier **README** détaillant :
     - Les étapes pour construire et exécuter les conteneurs.
     - Les configurations spécifiques à chaque environnement.
     - Les commandes pour tester les services.
     - Les bonnes pratiques appliquées.
3. **Fichier de logs des commits** :
   - Généré avec la commande :

```
git log --pretty=format:"%h %ad | %s%d [%an]" --date=short > logs_projet.txt
```

4. **Présentation finale** :
   - Démonstration de X minutes avec :
   - Présentation orale ou écrite expliquant les choix techniques, les difficultés rencontrées, et les solutions apportées.
   - Démonstration du fonctionnement de l'application.
   - Justifications techniques et gestion des problèmes rencontrés si applicable.

---

7. Critères d'Évaluation

Critère	Points
Création et optimisation des Dockerfile	6
Configuration des fichiers Docker Compose/swarm	5
Respect des bonnes pratiques de sécurité et d'optimisation	3
Qualité de la documentation	1
Tests et validation fonctionnelle	2
Total	**16**

---

8. Processus recommandé (non obligatoire)

- **Planification** :
  - Répartissez les tâches entre les membres de l'équipe.
  - Établissez un planning avec des objectifs intermédiaires.
- **Communication** :
  - Communiquez régulièrement sur l'avancement et les obstacles rencontrés.
- **Documentation continue** :
  - Documentez votre travail au fur et à mesure.
  - Gardez une trace des problèmes rencontrés et des solutions apportées.
- **Tests réguliers** :
  - Testez fréquemment pour détecter rapidement les problèmes.
  - N'attendez pas la fin du projet pour valider le fonctionnement.
- **Adoption des bonnes pratiques** :
  - Suivez les recommandations officielles pour Docker et Docker Compose.
  - Portez une attention particulière à la sécurité dès le début.
- **Technique** :
  - Essayer d'abord de faire le projet sans bonne pratiques, en mode développement et ajouter les multi-stages, environnements, au fur et a mesure.

---

9. Ressources

- **Documentation Docker** : https://docs.docker.com/
- **Bonnes Pratiques Dockerfile** : https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
- **Documentation Docker Compose** : https://docs.docker.com/compose/
- **Trivy pour les scans de sécurité** : https://aquasecurity.github.io/trivy/