TP Final : Déploiement Automatisé d'une Application 3-Tiers sur Cluster Kubernetes

Module : INF4052 - Virtualisation et Conteneurisation

Durée estimée : 6h (Projet)

Groupe : 3 ou 4 membres

1. Contexte et Objectifs

Vous êtes ingénieur DevOps dans une startup. L'équipe de développement vient de livrer une nouvelle application web "Fullstack" (Frontend + Backend + Base de données).

Votre mission est de :

1. **Conteneuriser** cette application pour garantir sa portabilité.
2. Mettre en place une **intégration continue (CI)** pour construire les images automatiquement.
3. Préparer une infrastructure de production composée de **2 Machines Virtuelles** simulant un cluster.
4. Automatiser le déploiement sur ce cluster via **Ansible**.

2. Architecture Cible

- **Source :** Gitlab (Code source).
- **Build :** Gitlab CI (Construction des images Docker).
- **Infrastructure :** 2 VMs locales (VirtualBox ou VMware) tournant sous Linux (Debian/Ubuntu).
    - *VM-Master* : Nœud maître Kubernetes (Control Plane).
    - *VM-Worker* : Nœud agent (Exécution des pods).
- **Orchestrateur :** Kubernetes (Distribution légère **K3s** recommandée).
- **Déploiement :** Script Ansible (lancé depuis votre poste de travail).

3. Consignes de Réalisation

Partie 1 : L'Application et la Conteneurisation (Docker)

Récupérez le code source d'une application "Demo" open-source. *Conseil : Utilisez l'application classique "Result/Vote" ou une simple stack MERN/PERN.*

- **Repo suggéré :** Forkez ce projet : https://github.com/dockersamples/example-voting-app (ne gardez que `vote`, `result` et `db` pour simplifier, ou utilisez votre propre stack Front/Back/DB).

Travail à faire :

1. Créez un `Dockerfile` optimisé (Multi-stage build) pour le Frontend.
2. Créez un `Dockerfile` pour le Backend.
3. Créez un fichier `docker-compose.local.yml` permettant de tester toute la stack en local sur votre machine (avec le réseau et les volumes pour la BDD).
4. **Livrable :** Le dépôt Git contenant les Dockerfiles et le docker-compose fonctionnel.

Partie 2 : Pipeline d'Intégration Continue (Gitlab CI)

Sur le Gitlab de l'école (ou gitlab.com si performances dégueu), configurez un pipeline `.gitlab-ci.yml`.

Travail à faire :

1. **Stage Build :** Le pipeline doit se déclencher à chaque push.
2. Il doit construire les images Docker du Front et du Back.
3. **Stage Push :** Il doit pousser ces images dans le **Gitlab Container Registry** associé à votre projet, avec le tag `:latest` et le tag du commit (`$CI_COMMIT_SHORT_SHA`).

Partie 3 : Infrastructure et Kubernetes

Vous devez simuler un environnement de production.

Travail à faire :

1. Installez **2 VMs** (Master et Worker) connectées en réseau (Bridge ou Host-only pour qu'elles se voient et que vous puissiez les pinger).
2. Installez un cluster Kubernetes léger (**K3s** est fortement recommandé pour sa simplicité, ou MicroK8s).
    - La *VM-Master* lance le serveur.
    - La *VM-Worker* rejoint le cluster.
3. Rédigez les fichiers manifestes Kubernetes (`k8s/`) :
    - `postgres-deployment.yaml` + `postgres-service.yaml` (ClusterIP).
    - `backend-deployment.yaml` + `backend-service.yaml` (ClusterIP).

- `frontend-deployment.yaml` + `frontend-service.yaml` (NodePort pour accès depuis l'hôte).

Partie 4 : Automatisation du Déploiement (Ansible)

C'est le cœur du sujet. Vous ne devez pas copier les fichiers YAML manuellement sur les serveurs.

Travail à faire : Écrivez un playbook Ansible `deploy.yml` et un inventaire `hosts.ini` qui effectue les actions suivantes (depuis votre machine hôte vers la VM Master) :

1. Se connecte à la *VM-Master*.
2. Copie les fichiers manifestes (`.yaml`) du dossier local vers la VM.
3. (Optionnel) Utilise `sed` ou le module `template` d'Ansible pour mettre à jour la version de l'image Docker dans les fichiers K8s (utiliser le tag du dernier commit).
4. Exécute la commande `kubectl apply -f <dossier>` sur la VM Master pour mettre à jour l'application.

4. Scénario de Démonstration (Sur le rapport final)

Pour valider le TP, vous devrez effectuer la démonstration suivante :

1. **Modification :** Vous changez une couleur ou un texte dans le code du Frontend sur votre machine.
2. **Push :** Vous faites un `git commit` et `git push`.
3. **CI :** On observe le pipeline Gitlab passer au vert et les nouvelles images apparaître dans le registre.
4. **Deploy :** Vous lancez la commande `ansible-playbook -i hosts.ini deploy.yml`.
5. **Vérification :** On ouvre le navigateur sur l'IP de la VM Worker (port défini dans le NodePort) et on voit la modification apparaître.

Livrables

Vous avez jusqu'au Dimanche 14 Décembre 2025 à 23h59 pour votre rendu complet de groupe à l'adresse milodigitalcreations@gmail.com.

Votre mail aura pour objet, et votre PDF également (nom de fichier) : **ESIEA-VIRTCONT-[Votre TD]-2025-NOM1-NOM2-NOM3-NOM4.pdf**

Le rendu se compose de deux éléments : le dépôt Git (contenant tout le code) et un **Rapport Technique au format PDF**.

Contenu du Rapport PDF

Ce document ne doit pas être un simple copier-coller de votre code (le code est déjà sur Git). Il doit synthétiser votre compréhension et votre démarche d'ingénieur.

Format : PDF, 5 à 10 pages maximum (hors annexes).

Le rapport doit impérativement contenir les sections suivantes :

1. Architecture et Choix Techniques

- Un **schéma d'architecture** montrant les interactions entre votre machine hôte, Gitlab, et les deux VMs (flux réseaux, ports ouverts).
- Justification rapide de vos choix (ex: quelle image de base Docker choisie et pourquoi ? Quelle stratégie de taggage dans la CI ?).

2. Démarche de Mise en Œuvre

Expliquez les grandes étapes de votre réalisation.

- Comment avez-vous optimisé vos images Docker (Multi-stage build) ?
- Comment avez-vous configuré la communication réseau entre les deux VMs pour Kubernetes ?
- Logique de votre playbook Ansible (idempotence, gestion des erreurs).

3. Difficultés Rencontrées et Solutions

C'est la section la plus importante pour l'évaluation. Ne dites pas "tout s'est bien passé". Décrivez les problèmes techniques réels rencontrés (ex: problèmes de droits, erreur de réseau entre les conteneurs, crash de K3s, indentation YAML, etc.).

- **Problème :** Description de l'erreur ou du blocage.
- **Analyse :** Comment avez-vous diagnostiqué la cause ?
- **Solution :** Quelle correction avez-vous appliquée ?

4. Usage de l'IA Générative (Transparence)

L'utilisation d'assistants IA (ChatGPT, Claude, Copilot, etc.) est autorisée. Dans cette section, détaillez honnêtement comment vous les avez utilisés.

- **Usage :** Avez-vous utilisé l'IA pour générer des squelettes de code ? Pour expliquer une erreur obscure ? Pour optimiser un script ?
- **Critique :** L'IA a-t-elle fait des erreurs que vous avez dû corriger ?
- *Note : L'honnêteté est valorisée. Dire "J'ai utilisé ChatGPT pour générer le script Ansible mais j'ai dû corriger les chemins de fichiers qui étaient faux" est une excellente réponse. Prétendre n'avoir rien utilisé alors que le code est générique sera pénalisant.*

5. Démonstration (Preuves de fonctionnement)

Cette section doit contenir des **captures d'écran commentées** prouvant que le TP est fonctionnel :

- **CI/CD :** Screenshot du pipeline Gitlab "Vert/Réussi" avec le détail des jobs. (ajoutez une image de pingouin et gagnez 1 point)
- **Registry :** Screenshot prouvant que les images sont bien arrivées dans le registre Gitlab.
- **Cluster :** Screenshot de la commande `kubectl get nodes` montrant le Master et le Worker en status `Ready`.
- **App :** Screenshot de votre navigateur affichant l'application finale modifiée.