TD1 : Principes DevOps

1 Intégration continue

On développe une application web de gestion de réservations, composée de :

- une API REST (backend en Python/Flask) ;
- une base de données relationnelle (PostgreSQL) ;
- une interface web (frontend en React).

L'équipe souhaite mettre en place une chaîne d'intégration continue (CI) permettant d'automatiser les tests, de valider les contributions via des Pull Requests.

1.1 Automatiser les tests

Expliquez quels types de tests peuvent être automatisés dans ce projet :

- tests unitaires ;
- tests d'intégration ;
- tests end-to-end (E2E).

Pour chaque type de test, proposez :

- un exemple concret adapté à ce projet (ex : test unitaire d'une fonction de calcul de disponibilité) ;
- le moment où il est pertinent d'exécuter ces tests dans un pipeline CI (ex : à chaque commit, avant le déploiement, seulement la nuit).

1.2 Pull Request

On suppose qu'un contributeur ouvre une Pull Request (PR) pour ajouter une nouvelle fonctionnalité au backend. Décrivez les étapes à suivre pour intégrer cette PR dans le projet, en précisant :

1) les vérifications automatiques à exécuter (tests, linting, build) ;

2) les validations humaines nécessaires (review par un pair, approbation) ;

3) les actions automatiques possibles après merge (déploiement sur un environnement de staging, mise à jour de la documentation).

2 Déploiement d'un moteur de transformation sur une infrastructure Spark

Contexte. On souhaite déployer une infrastructure basée sur **Apache Spark**, afin de réaliser des transformations de données en environnement distribué (passage de modèles de classes vers des modèles relationnels). L'assemblage cible de notre infrastructure comprend les éléments suivants :

- un **Spark Master**, responsable de la coordination des tâches ;
- trois **Spark Workers**, chargés de l'exécution des calculs distribués ;
- un service **SparkTE**, servant d'interface utilisateur pour la soumission des transformations ;
- une **base de données d'entrée** et une **base de données de sortie**, chacune possédant des réplicats.

2.1 Relation d'ordre partiel strict

Rappelez les trois propriétés d'une relation d'ordre partiel strict : (càd irréflexivité, antisymétrie, transitivité). Illustrez chacune de ces propriétés à l'aide d'un exemple simple (hors contexte Spark), par exemple avec des relations sur les nombres entiers.

2.2 Modèle de composants

Modélisez l'architecture décrite en un **diagramme de composants** :

- représentez les différents modules (Spark Master, Spark Workers, SparkTE, bases de données)
- identifiez les relations de dépendance entre eux
- justifiez vos choix de dépendances en termes de contraintes de déploiement.

2.3 Déployer une architecture Spark

En utilisant la notion d'ordre partiel strict, proposez un plan de déploiement dans les situations suivantes :

1) **Déploiement initial** avec 1 Master et 3 Workers ;

2) **Remplacement d'un Spark Worker** défectueux ;

3) **Ajout d'un gestionnaire YARN** pour la gestion des ressources. Pour info, YARN se déploie comme un spark-worker ;

4) **Changement des paramètres du Master** (redéploiement du service) ;

5) **Destruction complète** de l'infrastructure.

Pour chaque scénario :

- décrivez les relations de dépendances pertinentes ;
- indiquez un ordre de déploiement (ou de retrait) valide.

2.4 Utiliser l'Infrastructure-as-Code

Classez chacune des opérations suivantes, en fonction du type d'outil d'Infrastructure as Code (management d'application, provisionnement d'infrastructure, installation et configuration, orchestration) avec lequel l'effectuer :

1) Créer une machine virtuelle sur un cloud provider (par ex. AWS, GCP).

2) Construire une image Docker contenant Spark et ses dépendances.

3) Lancer un conteneur Spark Worker à partir d'une image préexistante.

4) Déployer un cluster Kubernetes pour héberger Spark.

5) Définir la taille et la zone de disponibilité des disques persistants pour la base de données.

6) Appliquer un script Ansible pour configurer la base de données (utilisateurs, droits).

7) Mettre en place une règle de firewall ouvrant un port pour Spark Master.

8) Définir dans Kubernetes un service exposant SparkTE via un LoadBalancer.

9) Modifier le fichier de configuration `spark-defaults.conf` pour changer les paramètres du Master.

10) Détruire toutes les ressources cloud associées au cluster (machines, volumes, réseaux).

Justifiez brièvement vos choix.