​Text

Description automatically generated

LOG3000 – INGÉNIERIE LOGICIELLE

**TP4**

Présenté à:

Mohammad Hamdaqa

Écrit par :

Cassy Charles (1947025)

Samy Checklat (1937812)

Remis le 17 Avril 2022

# 4.2. Questions

## **Q1. Dans chaque stratégie de déploiement ci-dessous, donnez la stratégie de déploiement appropriée qu'elle représente et un exemple concret.**

**Q2. Quelle est la relation entre OpenShift et Kubernetes?**

Kubernetes et OpenShift présentent tous deux une architecture robuste et évolutive qui permet le développement, le déploiement et la gestion d'applications rapides et à grande échelle. Ils fonctionnent tous deux sous la licence Apache 2.0. Red Hat® OpenShift® est une distribution Kubernetes, un produit logiciel commercialisé dérivé d'un projet open source. Red Hat OpenShift et Kubernetes sont tous deux des logiciels d'orchestration de conteneurs, mais Red Hat OpenShift est conditionné en tant que plate-forme open source d'entreprise en aval, ce qui signifie qu'il a subi des tests supplémentaires et qu'il contient des fonctionnalités supplémentaires qui ne sont pas disponibles dans le projet open source Kubernetes.

**Q3. Dans vos propres mots, qu'est-ce que les Kubernetes et pourquoi est-il utile pour DevOps?**

Kubernetes est une plateforme open source qui automatise les opérations de conteneurs Linux. Elle élimine une grande partie des processus manuels impliqués dans le déploiement et la mise à l'échelle des applications conteneurisées, ce qui veut dire qu'on peut regrouper des groupes d'hôtes exécutant des conteneurs Linux, et Kubernetes aide à gérer facilement et efficacement ces conteneurs.

L'une des principales raisons d'utiliser Kubernetes pour DevOps est qu'il réduit la charge de travail. Kubernetes améliore la qualité des processus DevOps avec la cohérence de l'environnement de développement, de test et de déploiement, la prise en charge de plusieurs frameworks, l'intégration d'une nouvelle application prend moins de temps, l’amélioration de la productivité et de l'efficacité des développeurs. Kubernetes facilite aussi la livraison rapide des logiciels avec une meilleure conformité, améliore la collaboration et la transparence au sein des équipes chargées de livrer le logiciel et réduit efficacement les coûts de développement et les risques de sécurité.

**Q4. Expliquez la différence entre la mise à l'échelle horizontale et la mise à l'échelle verticale.**

La mise à l'échelle horizontale fait référence à l'ajout de plusieurs périphériques ou nœuds informatiques au système pour améliorer les performances, tandis que la mise à l'échelle verticale correspond à l'ajout de ressources supplémentaires à un seul périphérique informatique afin d'améliorer les performances. [1]

**Q5. Qu'est-ce qu'un pod? Expliquez comment ils fonctionnent.**

Les pods sont les plus petites unités déployables dans Kubernetes. Comme l'indique la documentation officielle : "Un pod est un groupe d'un ou plusieurs conteneurs, avec des ressources de stockage/réseau partagées, et une spécification sur la façon d'exécuter les conteneurs." Ainsi, dans les termes les plus simples possibles, un pod est le mécanisme qui permet d'activer un conteneur dans Kubernetes.

Les pods sont un produit des contrôleurs qui sont responsables de la gestion des opérations au sein du système Kubernetes. Les contrôleurs facilitent le déploiement, la réplication et l'état général des pods dans un cluster. Le contrôleur est le délégant de la réplication des pods si un pod échoue. Il existe trois principaux types de contrôleurs, à savoir les jobs, les déploiements et les statefulSets. Les travaux sont de courte durée, c'est-à-dire qu'ils n'existent que jusqu'à ce qu'un travail soit terminé. [2]

**Q6. Qu'est-ce que “l'autoguérison d'application”?**

Un système auto-réparateur peut découvrir des erreurs dans son fonctionnement et se modifier sans intervention humaine, ce qui lui permet de retrouver un meilleur état de fonctionnement. Dans les applications typiques, les problèmes sont documentés dans un "journal des exceptions" pour un examen ultérieur. La plupart des problèmes sont mineurs et peuvent être ignorés. Les problèmes graves peuvent nécessiter l'arrêt de l'application (par exemple, l'impossibilité de se connecter à une base de données qui a été mise hors ligne).

En revanche, les applications auto réparatrices intègrent des éléments de conception qui permettent de résoudre les problèmes. Par exemple, les applications qui utilisent Akka organisent les éléments dans une hiérarchie et attribuent les problèmes d'un acteur à son superviseur. Un grand nombre de bibliothèques et de frameworks de ce type facilitent les applications qui s'autoréparent par conception. [3]

**Q7. Quel est le but du routage?**

**Q8. Dans l'interface du « portail d'apprentissage interactif », Sélectionnez deux scénarios: (1) Using the CLI to Manage Resource Objects (2) Transferring Files in and out of Containers Expliquez ce qui a été fait dans chaque scénario avec vos propres mots. Utilisez des captures d'écran pour appuyer votre réponse.**

**Q9. Maintenant, vous devez implémenter le déploiement bleu-vert avec le code source disponible sur GitHub.**

# 4.3. Question de rétroaction

**Nous travaillons à l’amélioration continue des travaux pratiques de LOG3000. Cette question peut être répondue très brièvement. Combien de temps avez-vous passé au travail pratique, en heures-personnes, en sachant que deux personnes travaillant pendant trois heures correspondent à six heures-personnes ? Est-ce que l'effort demandé pour ce laboratoire est adéquat ?**

Nous avons passé 8 heures-personnes sur ce travail pratique car la partie tutoriel nous a pris un peu de temps. L’effort demandé pour ce laboratoire est adéquat.