Résumé module 7 - Getting Started with Google Kubernetes Engine

Introduction aux conteneurs et à Kubernetes

Les conteneurs représentent une évolution majeure par rapport aux machines virtuelles traditionnelles. Ils sont beaucoup plus légers car ils ne transportent pas un système d'exploitation complet et peuvent être intégrés facilement avec le système sous-jacent. Cette approche permet aux développeurs de créer des applications plus performantes et évolutives, avec la garantie que le code fonctionnera de la même manière en local et en production.

Un **conteneur** n'est qu'une instance en cours d'exécution d'une image de conteneur. Cette image contient l'application et toutes ses dépendances. L'isolation des conteneurs repose sur plusieurs technologies Linux comme les *namespaces* et les *cgroups*. Il faut noter que les images de conteneurs sont éphémères , et donc tous les changements effectués pendant l'exécution sont perdus à l'arrêt du conteneur.

Google propose **Cloud Build** comme service géré pour construire des conteneurs. Ce service s'intègre avec Cloud IAM et peut récupérer le code source depuis différents dépôts comme Cloud Source Repositories, GitHub ou Bitbucket.

Kubernetes est une plateforme open-source qui permet d'orchestrer des charges de travail conteneurisées à grande échelle. Il facilite le déploiement, la gestion et la mise à l'échelle des applications sous forme de microservices. Le système fonctionne comme un ensemble d'APIs pour déployer des conteneurs sur un cluster de nœuds.

Google Kubernetes Engine (GKE)

GKE est un service Kubernetes géré et hébergé sur l'infrastructure Google. Son objectif principal est d'aider à déployer, gérer et faire évoluer les environnements Kubernetes pour les applications conteneurisées. GKE apporte plusieurs avantages : des systèmes d'exploitation optimisés pour une montée en charge rapide, la gestion automatique des configurations avec Autopilot, les mises à jour automatiques, la réparation automatique des nœuds défaillants, et l'intégration avec les autres services Google Cloud.

La différence principale avec Kubernetes standard, c'est que GKE gère tous les composants du plan de contrôle pour nous. On n'a qu'à envoyer nos requêtes API à une adresse IP exposée, et GKE s'occupe de toute l'infrastructure derrière.

Architecture et composants Kubernetes

L'architecture Kubernetes repose sur un modèle d'objets où chaque élément géré est représenté par un objet avec deux éléments importants : la spécification de l'état désiré et le statut actuel fourni par le plan de contrôle.

Un cluster se compose d'un plan de contrôle et de nœuds. Le plan de contrôle coordonne tout le cluster tandis que les nœuds exécutent les Pods. Les **Pods** sont les plus petits objets déployables dans Kubernetes et créent l'environnement où vivent les conteneurs. Chaque Pod reçoit une adresse IP unique et tous les conteneurs d'un même Pod partagent l'espace réseau.

Les composants principaux du plan de contrôle incluent :

- **kube-APIserver** qui accepte les commandes et gère l'état du cluster
- **etcd** qui stocke de manière fiable toute la configuration du cluster
- **kube-scheduler** qui programme les Pods sur les nœuds en fonction des contraintes
- **kube-controller-manager** qui maintient les objets Kubernetes via des boucles de contrôle

GKE Autopilot vs Standard

GKE propose deux modes de fonctionnement. Autopilot optimise la gestion avec une expérience "mains libres" ; moins de gestion, moins d'options de configuration, mais on ne paie que ce qu'on utilise. Il est optimisé pour la production avec des clusters créés selon les meilleures pratiques, une posture de sécurité renforcée, et une efficacité opérationnelle améliorée. Par contre, les options de configuration sont plus restrictives et on n'a pas d'accès SSH aux nœuds.

Le mode Standard offre plus de contrôle avec une gestion fine des permissions et de la configuration, mais nécessite plus de gestion manuelle et on paie pour toute l'infrastructure provisionnée.

Opérations avec kubectl

L'outil **kubectl** permet d'interagir avec les clusters Kubernetes. Il doit être configuré avec l'emplacement et les informations d'authentification du cluster. Pour GKE, on utilise la commande `gcloud container clusters get-credentials` pour configurer kubectl.

L'**introspection** est essentielle pour comprendre ce qui se passe dans le cluster. Les commandes principales sont `get` pour lister les ressources, `describe` pour obtenir des détails, `exec` pour exécuter des commandes dans les conteneurs, et `logs` pour voir les sorties des applications. Ces outils permettent de diagnostiquer les problèmes et de comprendre l'état des applications déployées.

Les Pods peuvent avoir différents statuts : Pending (en attente de programmation), Running (en cours d'exécution), Succeeded (terminé avec succès), Failed (échec), Unknown (état inconnu), ou CrashLoopBackOff (redémarrages en boucle suite à des erreurs).