Orchestration des conteneurs : quels cas d'usage et quelles solutions ?

On assiste depuis quelques années à une multiplication des solutions d'orchestration des conteneurs sur le marché, comme Kubernetes, Docker Swarm, AWS ECS, etc. sur lesquels nous reviendrons dans cet article.

Malgré les différences qu'on peut trouver entre ces solutions, elles répondent principalement à des besoins d'automatisation de la gestion des cycles de vies des conteneurs.

– **Le provisionning et le placement des conteneurs** : l'orchestrateur répartit et déploie les conteneurs sur les machines hôtes selon des besoins spécifiés en termes de mémoire et CPU.
– **Le monitoring** : une vue d'ensemble sur les métriques et les health-checks à la fois des conteneurs et des machines hôtes peut être portée par l'outil d'orchestration.
– **La gestion du failover des conteneurs et la scalabilité** : en cas d'indisponibilité d'une machine hôte par exemple, l'orchestrateur permet de redémarrer le conteneur sur une deuxième machine hôte. Cette scalabilité, en fonction de la solution d'orchestration utilisée, peut être manuelle ou automatique (autoscaling).
– **La gestion des mises à jour et rollbacks des conteneurs** : le principe du rolling update permet à l'orchestrateur d'assurer la mise à jour des conteneurs de manière successive et sans induire d'indisponibilité applicative. Pendant la phase de mise à jour d'un conteneur, les autres conteneurs disponibles sont exécutés.

En cas de besoin, l'orchestrateur permet également d'effectuer un retour-arrière sur la version précédente.

– **Le Service Discovery et la gestion du trafic réseau** : les conteneurs étant volatiles, les informations réseaux de chaque conteneur (comme l'adresse IP) sont variables. L'orchestrateur offre un niveau d'abstraction permettant de regrouper un ou plusieurs conteneurs, de leur allouer une adresse IP fixe et de l'exposer à d'autres conteneurs.

Orchestrateur de conteneurs : quelles sont les solutions du marché ?

Kubernetes, le leader des orchestrateurs

Développé initialement par Google en 2014, Kubernetes est une solution d'orchestration des conteneurs Open Source et gérée par Cloud Native Computing Foundation. Considérée comme étant la solution ayant le plus de notoriété sur le marché, Kubernetes dispose de la communauté la plus large parmi les outils d'orchestration du marché*.

Si Kubernetes offre un large panel de fonctionnalités natives d'orchestration (autoscaling, monitoring, load-balancing, etc.), sa prise en main reste toutefois complexe et nécessite une

montée en compétence à la fois sur la technologie et les principes des conteneurs, mais aussi sur les fonctionnalités propres à Kubernetes. A titre d'exemple, dans le cas d'une conteneurisation avec la technologie Docker, la maitrise à la fois de la CLI (Command Line Interface) de Kubernetes (kubectl) et de celle de Docker est nécessaire pour l'exécution des fonctionnalités offertes par Kubernetes.

Outre sa maturité et sa richesse fonctionnelle, Kubernetes permet d'orchestrer des conteneurs basés sur les technologies Docker, rkt ou encore des conteneurs basés sur les standards OCI (Open Container Initiative).

Docker Swarm, la solution native Docker

Créé en 2016 par Docker, l'orchestrateur Docker Swarm est la solution 100% Docker qui s'intègre exclusivement avec les conteneurs Docker. Ayant une notoriété et une popularité bien plus modeste que Kubernetes**, Docker Swarm reste tout de même l'un des principaux acteurs sur le marché des orchestrateurs.

A l'instar de Kubernetes, Docker Swarm met à disposition des fonctionnalités d'orchestration telles que la gestion des mises à jour, le service discovery ou le *scaling*. Toutefois, sur certains cas d'usage comme le Monitoring ou le Scaling, Docker Swarm possède quelques limites. En effet, contrairement à Kubernetes, on ne dispose pas de fonctionnalités natives de monitoring de bout en bout : une intégration d'un outil tiers comme Cadvisor devient dès lors requise.
Par ailleurs, le *scaling* manuel est possible via l'outil Docker swarm.
En revanche, contrairement à Kubernetes qui offre des fonctionnalités d'autoscaling nativement (Horizontal Pod Autoscaler pour la scalabilité des conteneurs et Cluster autoscaler pour la scalabilité des nœuds), la mise en place de l'autoscaling sur Docker Swarm nécessite l'intégration à des outils tiers et du développement supplémentaire.

Ayant une approche Docker-Centric, Docker Swarm est tout de même plus simple à appréhender que Kubernetes. Il s'intègre avec les outils Docker comme Docker Compose et Docker CLI. Par conséquent, il suffit de maitriser les concepts de la technologie Docker pour prendre en main plus aisément les fonctionnalités de Docker Swarm. L'effort en termes de montée en compétence sur la technologie, devient beaucoup moins lourd qu'une solution comme Kubernetes.

Les solutions packagées : AWS ECS, AWS Faregate, AWS EKS

Parmi les solutions d'orchestration des conteneurs du marché, on peut s'orienter vers l'alternative des solutions packagées proposées par plusieurs fournisseurs de services Cloud comme Amazon Web Services (AWS).

Ce choix peut être pertinent lorsque les applications sont conteneurisées exclusivement sur des environnements Cloud du fournisseur et que l'on souhaite par ailleurs utiliser les services clés en main qui sont déjà disponibles (Monitoring, Load Balancing, Stockage, etc.).

Prenons le cas d'AWS : **Amazon Elastic Container Service (ECS)** est le service d'orchestration des conteneurs proposé par Amazon. Plusieurs autres services AWS s'intègrent avec ECS, comme :

– Amazon CloudWatch qui permet le monitoring des conteneurs via la mise à disposition des

métriques (CPU et Mémoire). Ces métriques permettent de paramétrer les fonctionnalités d'autoscaling sur ECS ;

– Le service **Elastic Load Balancer** peut être utilisé pour la gestion du trafic réseau des conteneurs ;

– Elastic Block Storage, quant à lui, permet de définir les volumes de stockage Docker ;

– AWS Cloud Map couvre les fonctionnalités de Service Discovery des conteneurs ;

– Amazon CloudTrail assure la journalisation et trace les appels API effectués sur ECS.

Ces possibilités facilitent la prise en main de l'outil d'orchestration ECS, notamment par des utilisateurs qui maitrisent les services AWS.

Par ailleurs, ECS permet de provisionner les instances Amazon EC2, dimensionner les clusters de serveurs et de placer les conteneurs sur les instances, pour enfin exécuter les conteneurs et les orchestrer de manière centralisée. En d'autres termes, ECS offre une approche *« Do it yourself »* en permettant de garder la main et le contrôle sur l'infrastructure.

Une deuxième approche est possible avec la fonctionnalité **Fargate**, qui permet *a contrario* de se défaire de la gestion de l'infrastructure afin de se focaliser sur la conception et le développement des modules applicatifs. Fargate permet ainsi d'exécuter les applications conteneurisées selon les contraintes spécifiés (CPU, mémoire, réseau), tout en offrant un niveau d'abstraction sur les briques d'infrastructures et les instances EC2.

Outre les approches possibles via ECS, Amazon a également pensé à la communauté Kubernetes avec **Amazon Elastic Container Service for Kubernetes (EKS)**, qui propose tout simplement un service permettant l'utilisation des modules Kubernetes sur les environnements AWS.

Vous l'aurez compris : les solutions d'orchestration répondent à des besoins similaires, avec des approches différentes sur certains cas d'usage.

Le tableau de synthèse qui suit met en avant six principaux axes de comparaison.

	KUBERNETES	SWARM	AWS ECS
Provisionning et placement des conteneurs	Provisionning des briques sous-jacentes à la main de l'utilisateur	Provisionning des briques sous-jacentes à la main de l'utilisateur	2 possibilités : - Provisionning des briques sous-jacentes à la main de l'utilisateur (instances EC2 via ECS) - Utilisation de Faregate pour provisionning automatique des ressources sous-jacentes
Monitoring	Fonctionnalités natives disponibles (Horizontal Pod Autoscaler pour la scalabilité des conteneurs et Cluster autoscaler pour la scalabilité des nœuds)	Non disponible nativement Disponible via un outil tiers comme Cadvisor	Disponible via CloudWatch
Gestion du failover des conteneurs et la scalabilité	Autoscaling	Scaling manuel Autoscaling paramétrable via des outils tiers	Scaling manuel Autoscaling paramétrable via CloudWatch
Gestion des mises à jour et rollbacks des conteneurs	Fonctionnalités natives	Fonctionnalités natives	Fonctionnalités natives
Service Discovery et gestion du trafic réseau	Fonctionnalités natives	Fonctionnalités natives	Disponible via Cloud Map
Prise en main	Nécessite une montée en compétences dédiée Kubernetes et Docker	Nécessite une montée en compétences sur les technologies Docker	Nécessite des compétences sur Docker et les solutions AWS (CloudWatch, Cloud Map, etc.)

** Kubernetes : 85 k Commits et 2 300 contributeurs GitHub à date de novembre 2019*

*** Docker Swarm : 3,5 k Comits et 160 contributeurs GitHub à date de novembre 2019*