
The rt of
Kubernetes

Architecture Kubernetes Multi-Clusters

Il est possible que vous ressentiez le besoin de disposer de plusieurs clusters. Cela implique, ni plus ni moins, d'avoir plusieurs control plan, donc plusieurs cerveaux.

Le besoin le plus répandu pour la mise en place de multiples clusters est l'isolation des ressources par environnement. Il est essentiel de bien définir ce que chaque environnement inclut ou exclut. Les équipes applicatives auront besoin, au minimum, d'un environnement de développement, d'un environnement de test fonctionnel, d'un environnement de pré-production (staging), et, bien entendu, d'un environnement de production pour livrer le produit aux clients. Il est également envisageable d'ajouter un environnement d'intégration, permettant de vérifier la compatibilité entre les différentes applications et leurs dépendances, ou encore un environnement de performance, conçu pour tester la robustesse du produit dans des conditions identiques à celles de la production, sans impacter les autres environnements.

Un autre besoin réside dans l'isolation des données par zone géographique. Il serait, par exemple, non conforme de faire communiquer la charge de travail d'un cluster en Europe avec celle d'un cluster situé aux États-Unis. Pour répondre à ce

besoin, vous pouvez instancier un cluster par zone, agissant comme un isolant des données.

Maintenant que vous disposez de plusieurs clusters, il est probable que vous souhaitiez répartir vos charges de travail de manière optimisée entre ces différents environnements. Un outil particulièrement intéressant pour accomplir cela est Kubestellar. Cet outil présente l'avantage de ne nécessiter aucune modification de l'architecture de vos clusters Kubernetes existants. Il permet de centraliser la gestion de l'ensemble des ressources de vos clusters multiples, offrant ainsi une gestion simplifiée et unifiée depuis un point unique.

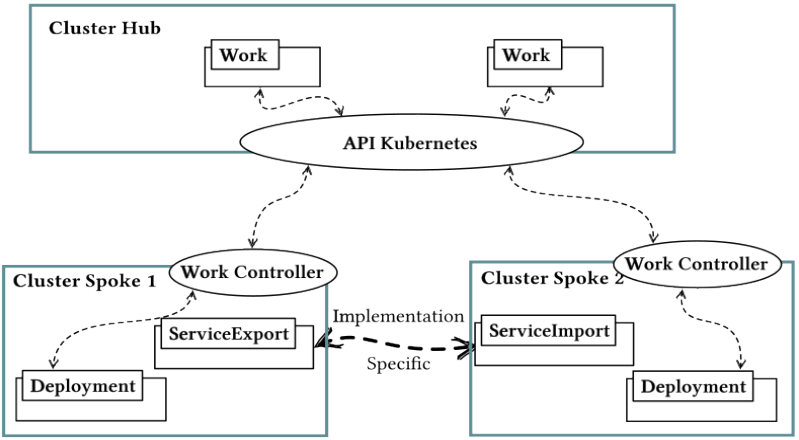
Il est également possible que vous envisagiez d'interconnecter vos clusters, notamment dans le but de mutualiser certains services de vos plateformes afin de réduire les coûts. Un Special Interest Group (SIG)[1] nommé Multicluster se penche sur cette problématique et propose des recommandations, des bonnes pratiques ainsi qu'une ligne de conduite pour y répondre. L'idée centrale consiste à fédérer plusieurs clusters Kubernetes afin de gérer les charges de travail de manière centralisée via une API Kubernetes unique. Dans cette approche, un cluster Kubernetes joue le rôle de « Hub » (cluster central) et communique avec des clusters dits « Spoke » (clusters satellites). Ce modèle de

1 Un site internet est accessible au grand public pour présenter les travaux en cours des différents groupes de travail Kubernetes, appelés SIG (Special Interest Groups). Par exemple, à l'adresse suivante : https://multicluster.sigs.k8s.io/, vous pouvez consulter les avancées et les projets en cours du SIG dédié au multicluster.

distribution, connu sous le nom de Hub and Spoke[2], est soutenu par plusieurs extensions de l'API Kubernetes.

Un Custom Resource Definition (CRD) Work représente une charge de travail constituée de plusieurs manifestes, ainsi que le cluster Spoke cible sur lequel elle doit être déployée. Un contrôleur, appelé Work Controller, prend en charge le déploiement de cette charge sur le cluster désigné. Quant à la gestion réseau des services, les CRD ServiceExport et ServiceImport font leur apparition, représentant respectivement l'exposition d'un service et son importation. Bien que leur implémentation, en termes conceptuels, reste relativement simple et proche des objets Service, ces objets sont spécifiquement adaptés aux environnements cloud.

2 Je vous recommande fort de vous familiariser avec le paradigme de Hub and Spoke, ou en Francais le paradigme d'architecture en étoile. Un très bon article sur wikipedia est disponible à l'adresse suivante : https://en.wikipedia.org/wiki/Spoke%E2%80%93hub_distribution_paradigm



La fédération de clusters est possible avec une architecture satellite. On peut déployer des objets Work dans l'API centrale, et les contrôleurs des clusters satellites prennent en charge leur déploiement. Cependant, les pods des clusters ne peuvent pas communiquer directement entre eux, sauf via les services exposés par ServiceImport et ServiceExport.

Cette stratégie multi-clusters comporte plusieurs avantages notables. Tout d'abord, il s'agit d'une solution développée par le projet Kubernetes, qui devrait donc continuer à évoluer et être maintenue dans les années à venir. Un autre avantage réside dans son architecture : elle permet d'être mise en place même après le déploiement de plusieurs clusters, pouvant ainsi être vue comme une amélioration naturelle de votre infrastructure. Cette architecture répond au besoin de distribution de charge sur plusieurs localisations avec des exigences d'interconnexion moindres comparé à une stratégie d'extension de clusters. La limite de cette architecture est que les Pods des différents clusters ne communiquent pas directement entre eux ; il est nécessaire de mettre en place le système ServiceExport /

ServiceImport afin qu'ils puissent interagir avec les services des autres clusters.

Il existe également d'autres solutions, certes moins "officielles", mais redoutablement efficaces. Si vous avez besoin d'interconnecter vos clusters afin que vos applications déployées sur un cluster puissent découvrir les services d'un autre comme s'ils faisaient partie du même cluster, Liqo est un outil minimaliste, open-source, et facile à prendre en main. Liqo permet également une certaine gestion de la répartition des charges de travail entre les différents clusters interconnectés. Il se distingue par sa simplicité de déploiement, puisqu'il ne requiert aucune modification des configurations existantes des clusters. Cependant, intégrer plusieurs clusters hétérogènes, notamment ceux gérés par différents fournisseurs Cloud, peut rendre sa configuration et sa gestion plus complexes. De plus, la latence des interconnexions réseau peut varier en fonction des distances géographiques entre les clusters, impactant ainsi les performances des applications distribuées.
Un autre projet, spécialisé dans l'interconnexion des clusters de manière sécurisée et performante, mais sans gestion de charge distribuée, est Submariner. Ce projet, désormais incubé par la CNCF, a été développé pour répondre aux défis de la connectivité réseau dans des environnements multi-cluster. Submariner vise les déploiements nécessitant une communication entre clusters tout en maintenant une stricte isolation réseau.
Submariner permet d'établir une connectivité transparente en configurant automatiquement des tunnels sécurisés et performants, permettant ainsi aux pods d'un cluster d'accéder

aux services d'un autre comme s'ils étaient sur le même réseau. Ce qui différencie Submariner, c'est son approche décentralisée qui ne nécessite aucun point de gestion centralisé pour orchestrer la connectivité, renforçant ainsi la disponibilité du réseau.
D'un point de vue technique, Submariner repose sur des tunnels VPN chiffrés, utilisant des technologies éprouvées comme IPsec ou WireGuard, garantissant ainsi une communication sécurisée entre les clusters Kubernetes. Cette approche facilite l'extension des réseaux tout en maintenant des niveaux de sécurité élevés.

Le concept de Plateforme en tant que Service (PaaS) que nous avons abordé dans ce livre est parfaitement compatible avec une architecture multi-cluster et vous laisse une certaine liberté quant à son implémentation. Cela se traduit concrètement par l'implémentation d'une plateforme par cluster, ou bien seulement d'une partie de la plateforme par cluster, tout en mutualisant certains services et outils sur d'autres clusters. Encore une fois, il est préférable de limiter le nombre d'outils et de services dans une plateforme au strict minimum, ainsi que le nombre d'instances associées. Bien que plusieurs outils de gestion permettent de gagner du temps dans l'exploitation des services, il convient de rester vigilant quant à la complexité supplémentaire qu'ils peuvent ajouter à la suite logicielle que vous devrez maîtriser. Je me répète, mais il est crucial d'adopter une approche pragmatique avec les PaaS et de se concentrer sur les niveaux de service que la plateforme fournit, plutôt que sur la multiplication des outils, des services et des fonctionnalités.