KUBERNETES

Maîtrisez l'orchestrateur
des infrastructures du futur

Chez le même éditeur

Docker - *Pratique des architectures à base de conteneurs*, P-Y. Cloux, T. Garlot, J. Kohler (2e édition) 336 pages, 2019.

Découvrir DevOps - *L'essentiel pour tous les métiers*, S. Goudeau, S. Metias (2e édition) 240 pages, 2018.

Mettre en œuvre DevOps - *Comment évoluer vers une DSI agile*, A. Sacquet, C. Rochefolle, (2e édition) 288 pages, 2018.

Scrum - *Pour une pratique vivante de l'agilité*, C. Aubry, (5e édition) 384 pages, 2018.

KUBERNETES

Maîtrisez l'orchestrateur des infrastructures du futur

Kelsey Hightower
Brendan Burns
Joe Beda

Traduit de l'anglais par Dominique Maniez

DUNOD

Authorized French translation of material from the English edition
of *Kubernetes : Up an d Running*
ISBN : 978-2-49-193567-5

Illustration de couverture : Tryaging-iStock

© Dunod, 2019
11 rue Paul Bert, 92240 Malakoff
www.dunod.com
ISBN 978-2-10-078940-5

Avant-propos

◆ *En forme de dédicace*

Kubernetes souhaiterait remercier tous les administrateurs système qui se sont réveillés à 3 heures du matin pour redémarrer un processus, mais aussi tous les développeurs qui ont mis leur code en production pour constater qu'il ne fonctionnait pas comme sur leur ordinateur portable, et puis encore tous les architectes système qui ont à tort lancé un test de charge sur un service de production en raison d'un nom d'hôte inutilisé qu'ils n'avaient pas mis à jour. Ce sont toutes ces souffrances, ces heures indues et ces erreurs étranges qui ont inspiré le développement de Kubernetes. Si l'on veut résumer notre propos en une seule phrase : Kubernetes a pour objectif de simplifier radicalement les tâches de création, de déploiement et de maintenance des systèmes distribués. Il a été inspiré par des décennies de pratique de conception de systèmes fiables et il a été élaboré à partir des usages de la base pour en faire une expérience, si ce n'est euphorique, tout du moins agréable. Nous espérons que vous allez apprécier ce livre !

◆ *À qui s'adresse ce livre ?*

Que vous soyez néophyte dans les systèmes distribués ou que vous déployiez des systèmes cloud-native depuis des années, les conteneurs et Kubernetes peuvent vous aider à réaliser des progrès en matière de vitesse, d'agilité, de fiabilité et d'efficacité. Cet ouvrage décrit l'orchestrateur de clusters Kubernetes et la façon dont ses outils et son API peuvent être utilisés pour améliorer le développement, la livraison et la maintenance des applications distribuées. Bien qu'aucune expérience antérieure avec Kubernetes ne soit requise, il est préférable d'être à l'aise avec la création et le déploiement des applications basées sur un serveur si vous voulez tirer le meilleur parti de ce livre. La familiarité avec des concepts comme l'équilibrage de charge et le stockage réseau sera utile, mais pas nécessaire. De la même manière, une expérience de Linux, des conteneurs Linux, et de Docker, n'est pas essentielle, mais elle vous aidera à profiter au maximum des ressources de cet ouvrage.

◆ *Pourquoi avons-nous écrit ce livre ?*

Nous avons été impliqués dans Kubernetes depuis son tout début. Ce fut vraiment extraordinaire d'assister à la transformation de cette technologie qui est passée d'une curiosité utilisée pour des expériences à une infrastructure cruciale de production qui alimente les applications à grande échelle dans des domaines variés qui vont de l'apprentissage automatique (*machine learning*) aux services en ligne. Quand cette transition s'est produite, il est devenu de plus en plus clair qu'un livre qui décrirait à la fois la façon d'utiliser les concepts de base de Kubernetes et les motivations sous-jacentes à l'élaboration de ces concepts serait une contribution importante pour le développement d'applications cloud-native. Nous espérons qu'à la lecture de ce livre, non seulement vous allez apprendre à créer des applications fiables et évolutives en vous appuyant sur Kubernetes, mais que vous allez aussi comprendre les défis majeurs des systèmes distribués qui ont conduit au développement de Kubernetes.

◆ *Un mot sur les applications cloud-native actuelles*

Des premiers langages de programmation, à la programmation orientée objet, en passant par le développement de la virtualisation et des infrastructures dans le cloud, l'histoire de l'informatique est une histoire du développement d'abstractions qui cachent la complexité et vous permettent de créer des applications toujours plus sophistiquées. Malgré cela, le développement d'applications fiables et évolutives est encore beaucoup plus difficile qu'il ne devrait l'être. Ces dernières années, les conteneurs et les API d'orchestration de conteneurs comme Kubernetes sont devenus une abstraction importante qui simplifie radicalement le développement de systèmes distribués fiables et évolutifs. Bien que les conteneurs et les orchestrateurs soient encore en phase de démocratisation, ils permettent déjà aux développeurs de construire et de déployer des applications avec une vitesse, une agilité et une fiabilité qui seraient passées pour de la science-fiction il y a seulement quelques années.

◆ *Organisation de ce livre*

Ce livre est organisé de la manière suivante. Le premier chapitre décrit les principaux avantages de Kubernetes sans rentrer trop profondément dans les détails. Si vous n'avez jamais utilisé Kubernetes, c'est l'occasion idéale pour commencer à comprendre pourquoi vous devriez lire le reste du livre.

Le chapitre suivant fournit une introduction détaillée aux conteneurs et au développement d'applications dans des conteneurs. Si vous n'avez encore jamais vraiment testé Docker, ce chapitre constitue une introduction utile. Si vous êtes déjà un expert Docker, ce sera sans doute une révision.

Le chapitre 3 couvre la procédure de déploiement de Kubernetes. Bien que la plus grande partie de ce livre se concentre sur la façon d'utiliser Kubernetes, vous devez disposer d'un cluster opérationnel avant de commencer à l'utiliser. Même si l'exécution d'un cluster pour la production dépasse la portée de ce livre, ce chapitre présente quelques méthodes simples pour créer un cluster afin que vous puissiez comprendre comment utiliser Kubernetes.

À partir du chapitre 5, nous abordons en détail le déploiement d'une application à l'aide de Kubernetes. Nous couvrons les *pods* (chapitre 5), les étiquettes et les annotations (chapitre 6), les services (chapitre 7), et les ReplicaSets (chapitre 8). Ces éléments constituent les bases fondamentales de ce dont vous aurez besoin pour déployer votre service dans Kubernetes.

Après ces chapitres, nous étudions quelques objets plus spécialisés dans Kubernetes : les DaemonSets (chapitre 9), les jobs (chapitre 10), les ConfigMaps et les secrets (chapitre 11). Bien que ces chapitres soient essentiels pour de nombreuses applications de production, vous pouvez les ignorer et y retourner plus tard, quand vous aurez acquis plus d'expérience et d'expertise, si vous commencez juste à découvrir Kubernetes.

Nous couvrons ensuite les déploiements (chapitre 12), qui mêlent le cycle de vie d'une application complète et l'intégration du stockage dans Kubernetes (chapitre 13). Enfin, nous concluons par quelques exemples illustrant comment développer et déployer des applications réelles dans Kubernetes.

◆ *Ressources en ligne*

Il est souhaitable d'installer Docker (*https://docker.com*) et de vous familiariser avec sa documentation si ce n'est déjà fait.

De la même manière, il est préférable d'installer l'outil en ligne de commande `kubectl` (https://kubernetes.io). Vous voudrez sans doute aussi vous abonner à la chaîne Slack Kubernetes (*http://slack.kubernetes.io*), où vous trouverez une grande communauté d'utilisateurs qui sont tout disposés à répondre à vos questions à presque n'importe quelle heure du jour et de la nuit.

Enfin, quand vous aurez bien progressé, vous pouvez participer au dépôt open source Kubernetes sur GitHub (*https://github.com/kubernetes/kubernetes*).

◆ *Conventions utilisées dans ce livre*

Les conventions typographiques suivantes sont utilisées dans ce livre :

Italique

Indique les nouveaux termes, les URL, les adresses électroniques, les noms de fichiers et les extensions de fichier.

`Police à espacement fixe`

Utilisée pour les listings de code des programmes, ainsi que dans les paragraphes pour faire référence à des éléments de programme comme des noms de variables ou de fonctions, des bases de données, des types de donnée, des variables d'environnement, des instructions et des mots-clés.

`Police à espacement fixe en gras`

Affiche les commandes ou tout autre texte qui doit être saisi littéralement par l'utilisateur.

`Police à espacement fixe en italique`

Affiche le texte qui doit être remplacé par des valeurs fournies par l'utilisateur ou par des valeurs déterminées par le contexte.

Cette icône représente une astuce, une suggestion ou une note.

Cette icône indique un avertissement ou une mise en garde.

◆ *Utilisation des exemples de code*

Les ressources supplémentaires (exemples de code, exercices, etc.) sont téléchargeables à l'adresse *https://github.com/kubernetes-up-and-Running/examples*.

Ce livre est conçu pour faciliter votre travail. En général, si un exemple de code est proposé avec ce livre, vous pouvez l'utiliser dans vos programmes et votre documentation.

Vous n'avez pas besoin de nous contacter pour une autorisation à moins que vous ne reproduisiez une partie importante du code. Par exemple, l'écriture d'un programme qui utilise plusieurs extraits de code de ce livre ne nécessite pas d'autorisation. En revanche, la vente ou la distribution d'un CD-ROM d'exemples extraits d'ouvrages de chez O'Reilly exige une autorisation. Si vous répondez à une question en citant ce livre et en citant un exemple de code, vous n'avez pas besoin d'une d'autorisation, mais si vous incorporez beaucoup d'exemples de code tirés de ce livre dans la documentation de votre produit, il vous faudra demander une autorisation.

Nous apprécions, sans que cela constitue une exigence, d'être référencés quand vous citez notre travail. Une référence inclut généralement le titre de l'ouvrage, son auteur, l'éditeur et le numéro ISBN. Par exemple : « *Kubernetes Up and Running* par Kelsey Hightower, Brendan Burns, et Joe Beda (O'Reilly), 2017, 978-1-491-93567-5 pour la version originale (*Kubernetes Maîtriser l'orchestrateur des infrastructures du futur*, Dunod, 2019 - 9782100789405 » pour la version en français).

Si vous pensez que votre utilisation d'exemples de code dépasse le cadre juridique de la courte citation, n'hésitez pas à nous contacter à permissions@oreilly.com.

Nous avons une page Web pour la version originale de ce livre en anglais, où nous répertorions les errata, les exemples et toutes les informations supplémentaires : *http://bit.ly/kubernetes-up-and-running* et sur *www.dunod.com* pour cette version en français.

Table des matières

Introduction

Kubernetes est un orchestrateur open source pour le déploiement d'applications en conteneurs. Kubernetes a été développé à l'origine par Google, qui s'est inspiré de sa longue expérience du déploiement de systèmes évolutifs et fiables dans des conteneurs via des API orientées application[1].

Mais Kubernetes ne s'est pas contenté d'exporter simplement la technologie développée par Google. Il a mûri pour devenir le produit d'une riche communauté open source en expansion. Cela signifie que Kubernetes est un produit qui est adapté non seulement aux besoins des entreprises présentes sur Internet, mais aussi à tous les développeurs cloud-native, qu'ils travaillent sur un cluster de Raspberry Pi ou sur une ferme de serveurs avec des ordinateurs à la pointe de la technologie. Kubernetes fournit le logiciel nécessaire pour construire et déployer avec succès des systèmes distribués fiables et évolutifs.

Vous vous demandez peut-être ce que nous entendons par « *systèmes distribués fiables et évolutifs* ». De plus en plus de services sont livrés sur le réseau via des API. Ces API sont souvent livrées par un *système distribué*, les différents éléments qui implémentent ces API fonctionnant sur différentes machines, connectées via le réseau et coordonnant leurs actions via des communications réseau. Dans la mesure où nous comptons de plus en plus sur ces API pour gérer tous les aspects de notre vie quotidienne (par exemple, trouver l'itinéraire vers l'hôpital le plus proche), ces systèmes doivent être extrêmement *fiables*. Ils ne peuvent pas se permettre de tomber en panne, même si une partie du système plante ou fonctionne mal. De la même manière, ils doivent rester *disponibles* même pendant les déploiements de logiciels ou les opérations de maintenance. Enfin, comme de plus en plus de monde est présent sur Internet et utilise de tels services, ils doivent être très *évolutifs* afin de pouvoir accroître leur capacité pour satisfaire une utilisation en augmentation continue sans avoir à modifier radicalement la conception du système distribué qui implémente les services.

1. Brendan Burns et al., "Borg, Omega, and Kubernetes: Lessons Learned from Three Container-Management Systems over a Decade," ACM Queue 14 (2016): 70–93, disponible à http://bit.ly/2vIrL4S.

Quelle que soit votre expérience en matière de conteneurs, de systèmes distribués, et de Kubernetes, nous pensons que ce livre vous permettra de tirer le meilleur parti de l'usage de Kubernetes.

Il existe de nombreuses raisons qui poussent les gens à utiliser des conteneurs et des API conteneur comme Kubernetes, mais nous estimons qu'ils cherchent tous à bénéficier au moins de l'un de ces avantages :

- ✓ Vitesse
- ✓ Évolutivité (du logiciel et des équipes)
- ✓ Abstraction de l'infrastructure
- ✓ Efficacité

Dans les paragraphes suivants, nous décrivons la manière dont Kubernetes peut vous aider à profiter de chacun de ces avantages.

1.1 VITESSE

La rapidité est le facteur clé aujourd'hui dans presque tous les développements informatiques. La nature du logiciel a changé et on est passé d'un programme vendu dans une boîte contenant un CD à des services basés sur le Web qui sont modifiés toutes les heures, ce qui signifie que ce qui fait la différence avec vos concurrents, c'est souvent la vitesse avec laquelle vous savez développer et déployer de nouveaux composants et de nouvelles fonctionnalités.

Il est cependant important de noter que cette rapidité n'est pas simplement définie en termes de vitesse brute. Même si vos utilisateurs sont toujours à la recherche d'améliorations itératives, ils resteront plus intéressés par un service extrêmement fiable. Par le passé, on tolérait qu'un service soit en maintenance à minuit tous les soirs, mais aujourd'hui, nos utilisateurs s'attendent à bénéficier d'une disponibilité constante, même si le logiciel qu'ils exécutent est mis à jour en permanence.

Par conséquent, la vitesse n'est pas mesurée en fonction du nombre de mises à jour que vous pouvez livrer en une heure ou en un jour, mais plutôt en fonction du nombre de choses que vous pouvez livrer tout en maintenant un service hautement disponible.

C'est dans ce but que les conteneurs et Kubernetes peuvent fournir les outils dont vous avez besoin pour réagir rapidement, tout en restant disponibles. Les concepts de base qui permettent cela sont l'immutabilité, la configuration déclarative, et les systèmes d'autoguérison en ligne. Ces idées sont toutes interdépendantes afin d'améliorer radicalement la rapidité avec laquelle vous pouvez déployer de manière fiable des logiciels.

1.1.1 La valeur de l'immutabilité

Les containers et Kubernetes encouragent les développeurs à créer des systèmes distribués qui respectent les principes d'infrastructure immutable. Avec une infrastructure immutable, une fois qu'un artefact est créé dans le système, les modifications de l'utilisateur ne peuvent pas le faire changer.

Traditionnellement, les ordinateurs et les systèmes logiciels sont considérés comme des infrastructures *mutables*. Avec une infrastructure mutable, les modifications sont appliquées en tant que mises à jour incrémentielles à un système existant. Une mise à

niveau du système via l'outil `apt-get update` est un bon exemple de mise à jour d'un système mutable. En exécutant `apt`, on télécharge séquentiellement tous les fichiers binaires mis à jour, on les copie par dessus des fichiers binaires plus anciens, et on effectue des mises à jour incrémentielles des fichiers de configuration. Avec un système mutable, l'état actuel de l'infrastructure n'est pas représenté comme un artefact unique, mais plutôt comme une accumulation de mises à jour et de modifications incrémentielles. Sur de nombreux systèmes, ces mises à jour incrémentielles proviennent non seulement des mises à niveau du système, mais aussi des modifications de l'exploitant.

En revanche, dans un système immutable, au lieu d'une série de mises à jour et de modifications incrémentielles, une image entièrement nouvelle et complète est créée, et la mise à jour remplace simplement en une seule opération la totalité de l'image par l'image la plus récente. Il n'y a pas de modifications incrémentielles. Comme vous pouvez l'imaginer, cela constitue un changement significatif dans l'univers plus traditionnel de la gestion des configurations.

Pour illustrer notre propos dans le monde des conteneurs, prenez l'exemple de deux manières différentes de mettre à niveau votre logiciel :

1. Vous pouvez vous connecter à un conteneur, exécuter une commande pour télécharger votre nouveau logiciel, supprimer l'ancien serveur et démarrer le nouveau.
2. Vous pouvez créer une nouvelle image de conteneur, la pousser dans un registre de conteneurs, supprimer le conteneur existant et en démarrer un nouveau.

À première vue, il peut sembler difficile de faire la différence entre ces deux approches. En quoi le fait de créer un nouveau conteneur peut-il améliorer la fiabilité ?

La différence principale est l'artefact que vous créez, et l'enregistrement de la façon dont vous l'avez créé. Ces enregistrements permettent d'identifier exactement les différences entre les versions et, si la nouvelle version a un souci, il est facile de déterminer ce qui a changé et comment résoudre le problème.

En outre, la construction d'une nouvelle image ne modifie pas une image existante, ce qui signifie que l'ancienne image est toujours présente, et qu'il est possible de l'utiliser rapidement pour une restauration si une erreur se produit. En revanche, une fois que vous copiez votre nouveau fichier binaire sur un fichier binaire existant, une telle restauration est presque impossible.

Les images de conteneurs immutables sont au cœur de tout ce que vous allez créer dans Kubernetes. Il est possible de changer grâce à des instructions les conteneurs en cours d'exécution, mais il s'agit là d'un anti-modèle à n'employer que dans les cas extrêmes où il n'y a pas d'autres options (par exemple, si c'est la seule façon de réparer temporairement un système critique de production). Et même dans ce cas-là, les modifications doivent aussi être enregistrées par le biais d'une mise à jour de configuration déclarative à un moment ultérieur, quand la crise est terminée.

1.1.2 Configuration déclarative

L'immutabilité s'étend au-delà des conteneurs en cours d'exécution dans votre cluster et s'applique aussi à la façon dont vous décrivez votre application à Kubernetes. Tout élément dans Kubernetes est un *objet de configuration déclarative* qui représente l'état désiré du système. C'est le travail de Kubernetes de s'assurer que l'état réel du monde correspond à cet état désiré.