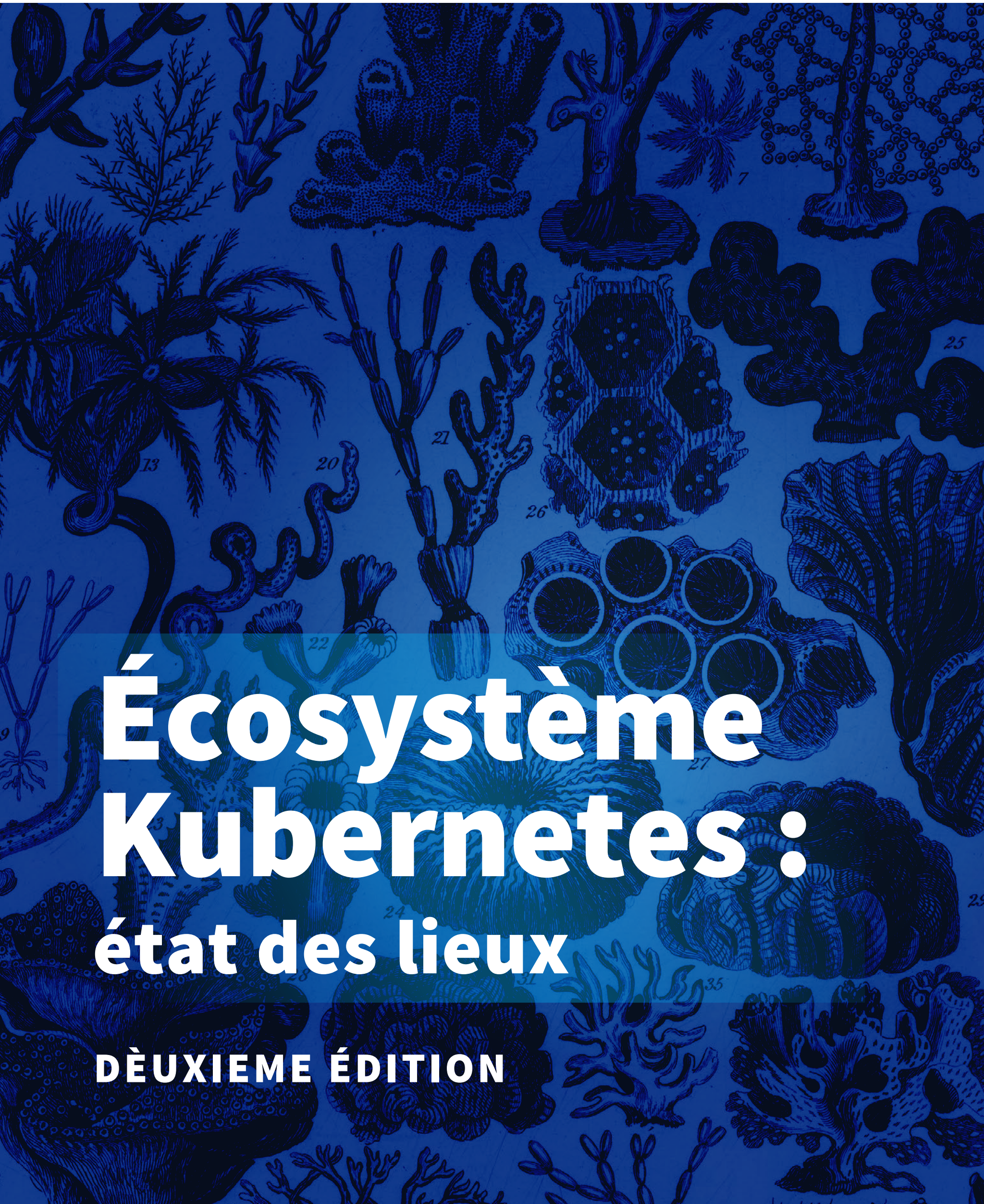
THENEWSTACK



Écosystème Kubernetes : état des lieux

DÈUXIEME ÉDITION

The New Stack

Écosystème Kubernetes : état des lieux, deuxième édition

Alex Williams, Fondateur et Directeur de publication

Équipe de l'eBook :

Gabriel H. Dinh, Producteur exécutif

Janakiram MSV (Janakiram & Associates), Auteur

Jonathan Gold (Container Solutions), Rédacteur technique

Judy Williams, Réviseuse

Lawrence Hecht, Contributeur en tant que chargé d'étude

Libby Clark, Directrice du marketing et de la rédaction

Richard MacManus, Relecteur de l'eBook

Sebastien Goasguen (TriggerMesh), Rédacteur technique

Équipe de soutien :

B. Cameron Gain, Correspondant UE

Benjamin Ball, Directeur des ventes et de la gestion de comptes

Colleen Coll, Responsable du marketing et des opérations médias numériques

Eddie Rogers, Producteur médias numériques

Jennifer Riggins, Correspondante Royaume-Uni

Joab Jackson, Rédacteur en chef

Michelle Maher, Secrétaire de rédaction

20210205-fr

Sommaire

Sponsors et partenaires 4

Crédits 6

Introduction 7

Kubernetes : le système d'exploitation pour le cloud 11

- Déploiements : la taille compte 14
- Orchestration : définition 17
- Architecture Kubernetes 20
- Kubernetes, clé de voûte de l'informatique web-scale 35
- Adopter Kubernetes 36
- Kubernetes, plan de contrôle universel 38

Architectures en self-service et opérateur Kubernetes pour Cassandra 41

L'observabilité de l'IA pour réduire la complexité de Kubernetes 42

Comment adapter des applications data-centric aux architectures Kubernetes 43

Mapper l'écosystème Kubernetes 44

- L'essor du cloud-native et du CaaS 45
- Caractéristiques clés d'une plateforme de gestion de containers 46
- Tour d'horizon des composants d'une plateforme de gestion de containers 47
- Plateforme de gestion de containers d'entreprise 69
- Plateforme managée de gestion de containers 72
- Adoption des offres Kubernetes de fournisseurs cloud 74

Kubernetes a évolué. Qu'en est-il de votre sécurité ? 76

Comment éliminer la fatigue liée à la complexité de Kubernetes 77

Kelsey Hightower sur son parcours personnel avec Kubernetes 78

Conclusion 79

Avertissement 82

Sponsors et partenaires

Nous tenons à remercier nos sponsors pour leur soutien sur cet eBook :

DataStax offre une plateforme de données NoSQL cloud-native, ultradisponible et hautement évolutive basée sur Apache Cassandra. Ainsi, il permet aux utilisateurs et aux entreprises d'exécuter des données sur n'importe quel cloud à très grande échelle, sans interruption de service ni enfermement contractuel.

Dynatrace est le leader de l'intelligence logicielle conçue pour le cloud d'entreprise. Il offre la seule plateforme d'intelligence complète 100 % automatisée et pilotée par IA, capable de fournir des éclairages approfondis sur les écosystèmes de cloud hybride dynamiques et web-scale. C'est pourquoi les plus grandes marques font appel à Dynatrace pour déployer des expériences utilisateurs irréprochables.

La plateforme de déploiement logiciel Harness automatise l'ensemble du processus CI/CD, ce qui permet aux ingénieurs de développer, de tester, de déployer et de vérifier leurs applications rapidement et en toute sécurité. Harness utilise le machine learning pour vous protéger en cas de défaillance de vos déploiements. La livraison de logiciels n'a jamais été aussi simple.

Hewlett Packard Enterprise

HPE Ezmeral vous aide à réaffecter du temps et des ressources des opérations IT vers l'innovation afin d'impulser vos projets de transformation numérique. Modernisez et sécurisez vos applications. Simplifiez vos opérations. Et libérez tout le potentiel de vos données pour passer à l'action.

THENEWSTACK

KubeCon | CloudNativeCon

Les conférences KubeCon + CloudNativeCon réunissent des adoptants et des technologues pour développer leurs connaissances et faire progresser l'informatique cloud-native. Parmi les participants à ces évènements indépendants figurent des spécialistes du domaine et des acteurs clés de projets bien connus comme Kubernetes, Prometheus, Envoy, CoreDNS, containerd et d'autres encore. KubeCon + CloudNativeCon sont organisées par la Cloud Native Computing Foundation (CNCF).

PRISMA™ BY PALO ALTO NETWORKS

Prisma facilite la migration vers le cloud en alliant le contrôle des accès à une protection intégrale des données, applications, hôtes, containers et ressources sans serveur. La suite de sécurité cloud la plus complète du marché aide les entreprises à passer à la vitesse supérieure grâce à la visibilité sur les risques et à la sécurité continue.

Merci également à nos partenaires :

Container Solutions

Containers Solutions est un fournisseur de services professionnels, spécialiste des transformations cloud-native. Il crée des synergies entre vos technologies, votre culture et votre stratégie afin d'optimiser votre transition. Container Solutions possède des bureaux à Amsterdam, Londres, Berlin, Montréal, Prague, Varsovie et Zurich.

TRIGGERMESH CLOUD NATIVE INTEGRATION PLATFORM

TriggerMesh propose une plateforme d'intégration cloud-native en temps réel qui vous permet d'interconnecter des services afin d'automatiser vos workflows et de fluidifier l'information dans votre organisation. Ainsi, il permet aux développeurs en entreprise de créer des applications basées sur les évènements et composées de services de multiples fournisseurs cloud et systèmes sur site.

Crédits

Janakiram MSV est l'auteur de cet eBook. Analyste en chef de Janakiram & Associates, il est aussi professeur adjoint à l'International Institute of Information Technology. Janakiram MSV est également titulaire de plusieurs certifications : Google Qualified Cloud Developer, Amazon Certified Solution Architect, Amazon Certified Developer, Amazon Certified SysOps Administrator et Microsoft Certified Azure Professional. Ambassadeur de la Cloud Native Computing Foundation, il fait partie des premiers administrateurs et développeurs d'applications Kubernetes certifiés. Auparavant, il a travaillé pour Microsoft, AWS, Gigaom Research et Alcatel-Lucent.

Lawrence Hecht a contribué à cet eBook en tant que chargé d'étude. Il produit des rapports d'étude sur les technologies de l'information depuis 17 ans. Auparavant, il était responsable des enquêtes « Voice of the Customer » de 451 Research et de TheInfoPro sur des marchés IT B2B comme le cloud computing, l'analytique de données et la sécurité informatique. En 1999, Lawrence Hecht a créé l'IPPN (Internet Public Policy Network), un réseau d'experts qui fournissait des études personnalisées, des livres blancs et des conseils sur des questions de politique publique liées aux technologies. Il travaille avec des rédacteurs afin de transformer les données d'études primaires et secondaires en informations exploitables. Actuellement, il met ses compétences au service de leaders d'opinion et de programmes de sensibilisation des pairs. Lawrence Hecht est titulaire d'un master en politique publique de l'université de Georgetown et d'un bachelor de la Rutgers University.

Introduction

Il y a un peu plus de cinq ans en juin 2015, le nom et le concept de Kubernetes, un projet tout droit sorti de Google plus tôt cette année-là, étaient encore peu connus.

Le nom Kubernetes vient d'un terme en grec ancien qui désigne le timonier ou le barreur d'un navire. Cette architecture technologique témoignait alors d'un changement enclenché bien avant 2015, avec l'essor des géants du web. Docker avait encore le vent en poupe, mais la communauté commençait à réfléchir à l'orchestration des containers. En juin 2015, une nouvelle organisation baptisée Cloud Native Computing Foundation (CNCF) est née et a repris le projet Kubernetes.

Les architectures scale-out avaient fait leur entrée. Docker était le tout premier projet open source de grande ampleur. Kubernetes lui a volé la vedette, au point de devenir un véritable mouvement en quelques années, à la croisée de l'informatique distribuée, des communautés open source et des architectures applicatives avancées. Il s'est vite imposé comme l'un des projets open source les plus importants de ces 20 dernières années. Certains le considéraient même comme la version cloud de Linux.

En 2017, pour documenter ce phénomène, The New Stack a publié son premier eBook sur la communauté Kubernetes et son écosystème cloud-native.

Depuis 2018, la communauté a évolué et s'est développée si vite que notre première édition commençait à dater. Il était donc temps de la mettre à jour et de dresser un nouvel état des lieux de Kubernetes.

D'après une enquête de 2019, près de 80 % des entreprises interrogées par la CNCF utilisent Kubernetes en production.

Kubernetes et les technologies cloud-native ont donc conquis un large public. Pour preuve la conférence KubeCon + CloudNativeCon, dont la fréquentation est passée de 1 000 personnes en 2016 à 4 000 personnes lors de son édition nord-américaine en 2017. L'automne dernier, KubeCon à San Diego comptait plus de 10 000 participants. En 2020, la conférence qui se tiendra en ligne pourrait bien battre des records de fréquentation.

À l'heure où nous publions ces lignes, le télétravail est devenu la norme et la Covid-19 contraint la communauté open source à s'en tenir au virtuel. Malgré cela, la croissance de Kubernetes se poursuit… de même que la consolidation de son écosystème.

Les 570 entreprises membres de la CNCF pèsent pas moins de 17 800 milliards de dollars. Le montant cumulé de ces capitalisations boursières reflète l'étendue des investissements dans la communauté. L'écosystème a également été le théâtre d'acquisitions notables. En janvier 2018, CoreOS, entreprise logicielle derrière le runtime de containers Tectonic, a été rachetée par Red Hat pour 250 millions de dollars. À la fin de l'année 2019, VMware s'est payé Heptio, l'entreprise fondée par Joe Beda et Craig McLuckie, deux des leaders de l'équipe de Google ayant développé Kubernetes – avec Brendan Burns. Ce dernier avait lui aussi quitté Google à l'époque, pour rejoindre Microsoft. Le rachat d'Heptio aurait coûté 450 millions de dollars à VMware. En mai 2019, Palo Alto Networks a fait l'acquisition de Twistlock, un éditeur de solutions de sécurité pour les containers, pour un montant de 410 millions de dollars. Le même mois, VMware a racheté Bitnami. Cette année-là, VMware a également annoncé l'acquisition de Pivotal, qui a conduit à la commercialisation de Cloud Foundry, la plateforme PaaS open source. Enfin, en juillet 2020, juste avant la publication de cet eBook, SUSE a racheté Rancher Labs pour 600 millions de dollars.

Parallèlement à la maturation du modèle d'architecture basée sur les microservices, l'expansion de la communauté Kubernetes a contribué au développement des connaissances sur les technologies d'orchestration de containers et, plus généralement, sur l'informatique distribuée.

Aujourd'hui, il est généralement entendu que Kubernetes nécessite un pool sous-jacent de ressources serveurs, réseau et de stockage exécutables via une infrastructure sur site ou un service cloud. Toutefois, les débutants comme les utilisateurs plus chevronnés gagneront à se voir expliquer l'architecture sous-jacente de base.

Dans cet eBook, The New Stack récapitule les connaissances essentielles sur Kubernetes. Réseau, stockage, nœuds worker, plans de contrôle, découverte de services… l'objectif est de vous aider à maîtriser tous les concepts de la plateforme. Nous examinerons les schémas d'adoption de Kubernetes, sa capacité à impulser une informatique web-scale et son rôle d'élément universel

des infrastructures d'entreprise.

Le second chapitre de cet eBook traite des plateformes de gestion de containers. Aujourd'hui, les éditeurs de logiciels et les services cloud proposent une grande variété de plateformes. Mais toutes ont un point commun : Kubernetes. Nous vous expliquerons en quoi Kubernetes s'impose comme l'architecture sous-jacente des plateformes destinées aux data centers d'entreprise, aux services cloud et à une approche hybride – sans oublier la périphérie réseau.

Également au sommaire de ce second chapitre : le rôle des technologies cloud-native sur les plateformes de containers. Nous verrons que les technologies cloud-native sont essentiellement des technologies de containers, adaptées à l'infrastructure containerisée et, surtout, Kubernetes.

Les microservices reposent sur des composants qui s'exécutent à travers de multiples containers. Mais comment gérer ces applications ? C'est là l'une des grandes questions qui continuent de tarauder les développeurs. Les applications sont un ensemble de composants. Or, la difficulté tient dans l'intégration de ces derniers, non seulement pour donner vie à l'application, mais aussi pour gérer le service tout au long de son cycle de vie.

Pour Janakiram MSV, l'auteur de cet eBook, le modèle de containers sous forme de service (CaaS) définit la stack cloud-native. Seulement voilà, il est complexe et difficile à gérer. De fait, une architecture Kubernetes en production s'apparente à une ruche. Au milieu du chaos, des pratiques d'ingénierie permettent aux microservices de s'exécuter dans des containers orchestrés par Kubernetes. Les entreprises qui saisissent cette réalité parviennent à développer et gérer des microservices à grande échelle. Leur croissance est assurée. Toutefois, la plupart des organisations font encore leurs premiers pas dans cet univers.

Cela résume bien le défi que la communauté Kubernetes doit relever après des débuts prometteurs, afin de répondre non seulement aux besoins des opérations à grande échelle, mais aussi à ceux des petites et moyennes entreprises.

La culture continue de poser problème. Les développeurs doivent pouvoir packager facilement les applications et il est temps d'éclaircir le rôle de chacun dans la configuration des services. Par ailleurs, les politiques de sécurité reposent encore largement sur les pratiques traditionnelles.

Pour Kubernetes et les éléments les moins matures de son écosystème, une marge de progression subsiste. Si Kubernetes constitue désormais le plan de contrôle incontesté, la question du plan de données reste globalement inexplorée. Bien que le maillage des services permette d'améliorer la gestion du trafic, la sécurité et l'observabilité, la prise en main n'est toujours pas une mince affaire.

Mais ce sont les concepts associés à l'observabilité dont nous attendons le plus. Les fonctions de surveillance convenaient parfaitement aux architectures d'entreprise sur site. Elles alimentaient une approche réactive. Mais cela ne suffit plus. Aujourd'hui, les entreprises doivent observer, anticiper et trouver des réponses.

Merci d'avoir téléchargé cet eBook et n'hésitez pas à nous contacter pour toute question.

Alex Williams

Fondateur et Directeur de publication

The New Stack

@alexwilliams

alex@thenewstack.io

CHAPITRE 1

Kubernetes : le système d'exploitation pour le cloud

Ces dix dernières années, le marché de l'infrastructure IT a été rythmé par plusieurs bouleversements, à commencer par l'arrivée de l'IaaS (Infrastructure as a Service) : un modèle de cloud computing qui a révolutionné le provisionnement et la consommation des ressources de calcul, de stockage et de réseau des entreprises. Plus tard, nous observions l'émergence des containers, des plateformes de gestion dédiées et des offres CaaS (Containers as a Service) dans le cloud.

Les services d'infrastructure cloud comme Amazon EC2, Azure Virtual Machines et Google Compute Engine se sont mis à fournir un modèle à la demande via un provisionnement de ressources basé sur le self-service et la programmation. Les entreprises disposaient ainsi d'une évolutivité à la fois rapide et optimisée.

En 2013, Docker, Inc. a présenté Docker, une plateforme légère de virtualisation au niveau du système d'exploitation (OS), basée sur les containers Linux. Exploitant une fonctionnalité inhérente à Linux, Docker permettait d'exécuter plusieurs applications isolées au sein du même système d'exploitation.

Traditionnellement, la virtualisation offrait la possibilité d'exécuter plusieurs systèmes d'exploitation au sein du même OS. Dans ce contexte, les hyperviseurs – des composants logiciels dédiés à la gestion des machines virtuelles (VM) – s'occupent du partitionnement et de la stricte isolation des VM. Les containers Linux assurent quant à eux la virtualisation au niveau de l'OS, ce qui permet d'exécuter sur un hôte plusieurs processus isolés (containers) à partir du même noyau Linux. Le kernel (ou noyau), qui exploite notamment les cgroups pour contrôler les ressources et les espaces de noms pour isoler les processus, gère également la priorité entre les composants comme

le processeur (CPU), la mémoire, les E/S de bloc et le réseau, le tout sans avoir à lancer de machines virtuelles.

En effet, contrairement aux VM, les containers ne dépendent pas d'un hyperviseur, puisqu'ils partagent les services du système d'exploitation sous-jacent au niveau du noyau. Les containers Linux sont plus petits, démarrent rapidement et offrent une montée en capacité efficace. Plus important encore, ils ont l'avantage d'être portables. Ainsi, un container bâti sur Ubuntu peut être directement déployé sur Red Hat, sans la moindre modification. De leur côté, les administrateurs démarrent les applications containerisées en quelques millièmes de seconde et les adaptent à des centaines d'instances en un rien de temps.

Même si les containers faisaient partie intégrante des systèmes UNIX modernes, via Linux (LXC), FreeBSD (Jails) ou encore Solaris (Zones), c'est bien Docker, Inc. qui a rendu cette technologie accessible afin de révolutionner le développement et le déploiement d'applications.

Ainsi, l'arrivée de l'IaaS et des containers dans le cloud public allait offrir aux entreprises une agilité et une évolutivité sans précédent. Le lancement de dizaines, voire de centaines de containers par machine virtuelle permettait d'exploiter au maximum l'utilisation des ressources CPU et de la mémoire. Aussi, le déploiement de containers dans le cloud public rendait possible et abordable l'exécution de workloads « web-scale », c'est-à-dire à l'échelle du web. Le duo IaaS-containers est donc devenu la formule gagnante des start-up du web.

Au fil du temps, malgré l'environnement d'exécution et les outils fournis par Docker, Inc. pour gérer le cycle de vie des containers, les acteurs du secteur ont compris qu'ils avaient besoin d'une plateforme dédiée pour faire face aux centaines de containers exécutés sur autant de machines virtuelles. Ainsi sont apparus Docker Swarm, développé par Docker, Inc., et DC/OS de D2iQ (anciennement Mesosphere).

Bien avant que cette technologie ne gagne en popularité chez les développeurs, Google exécutait déjà certains de ses principaux services web dans des containers Linux. Selon Joe Beda (l'un des créateurs de Kubernetes), qui s'est exprimé sur le sujet au cours d'une intervention à la GlueCon 2014, Google lançait ainsi plus de deux milliards de containers par semaine. Son secret ? Borg, un outil interne de gestion des data centers.

C'est donc en juin 2014 que Google présente Kubernetes : sa plateforme open source de gestion de containers à grande échelle. Le géant de Mountain View a développé cette nouvelle solution en reprenant les meilleurs concepts de Borg et en corrigeant les divers problèmes soulignés par les utilisateurs au fil des années.

En 2015, Kubernetes 1.0 est soumis à la Linux Foundation, qui forme la Cloud Native Computing Foundation (CNCF) dans la foulée pour gérer et gouverner le projet. Aujourd'hui, la CNCF est dépositaire de plusieurs initiatives open source liées aux containers, dont containerd, Envoy ou encore Prometheus.

Et Docker dans tout cela ? L'ambition de Docker était de simplifier le développement d'applications modernes. Après avoir installé le Docker Engine sur leur station de travail, les développeurs utilisaient les API et outils Docker pour gérer le cycle de vie des applications containerisées. Docker a par la suite introduit Compose et Swarm, deux extensions du moteur Docker de base permettant de déployer et de gérer les workloads multicontainers sur plusieurs nœuds/machines à l'aide du workflow et de la chaîne d'outils connus.

Mais Google est allé plus loin en autorisant l'exécution à grande échelle de différents types de workloads containerisés sur sa plateforme d'orchestration : l'extensibilité, l'évolutivité et la diversité des environnements de déploiement de Kubernetes lui ont permis de s'imposer rapidement chez les développeurs et les opérateurs.

Apache Mesos, un projet open source développé par l'université de Californie à Berkeley, est l'une des premières architectures IT distribuées pour gérer les workloads applicatifs sur les clusters informatiques. Pourtant, celle-ci a peu à peu été délaissée par le secteur et la communauté open source au profit de l'écosystème Kubernetes. En effet, bien que Mesos soit un système distribué mature, il se prête davantage aux systèmes à grande échelle avec plusieurs centaines de nœuds. Il faut savoir que ce cluster manager a d'abord été conçu pour les applications distribuées basées sur Apache Hadoop, Apache Spark et Kafka. La technologie d'orchestration des containers, quant à elle, est arrivée bien plus tard avec le plug-in Marathon. À l'inverse, Kubernetes est un outil spécialement prévu pour la containerisation, qui s'exécute avec autant de simplicité et de flexibilité sur toutes les tailles de clusters.

Docker, Inc. a d'ailleurs elle-même adopté l'ochestrateur Kubernetes en l'intégrant à Docker Desktop et Docker Enterprise. En novembre 2019, Mirantis, une entreprise spécialisée dans l'infrastructure OpenStack, rachète la branche Enterprise de Docker et hérite donc de ses offres Kubernetes as a Service (KaaS).

Grâce à sa simplicité, son accessibilité et son évolutivité, Kubernetes s'est imposé comme la première plateforme de gestion de containers. Ce projet open source connaît en effet l'une des croissances les plus rapides de l'histoire de l'informatique. Par ailleurs, tandis que les applications modernes et greenfield (comprenez « entièrement nouvelles ») se tournent de plus en plus vers l'orchestrateur de containers, on observe l'émergence de plusieurs plateformes de gestion dédiées telles que Google Anthos, Red Hat OpenShift et VMware Tanzu. Cela se traduit aussi par le renforcement des offres CaaS managées dans le cloud public, parmi lesquelles s'inscrivent Amazon Elastic Kubernetes Service, Azure Kubernetes Service et Google Kubernetes Engine.

Déploiements : la taille compte

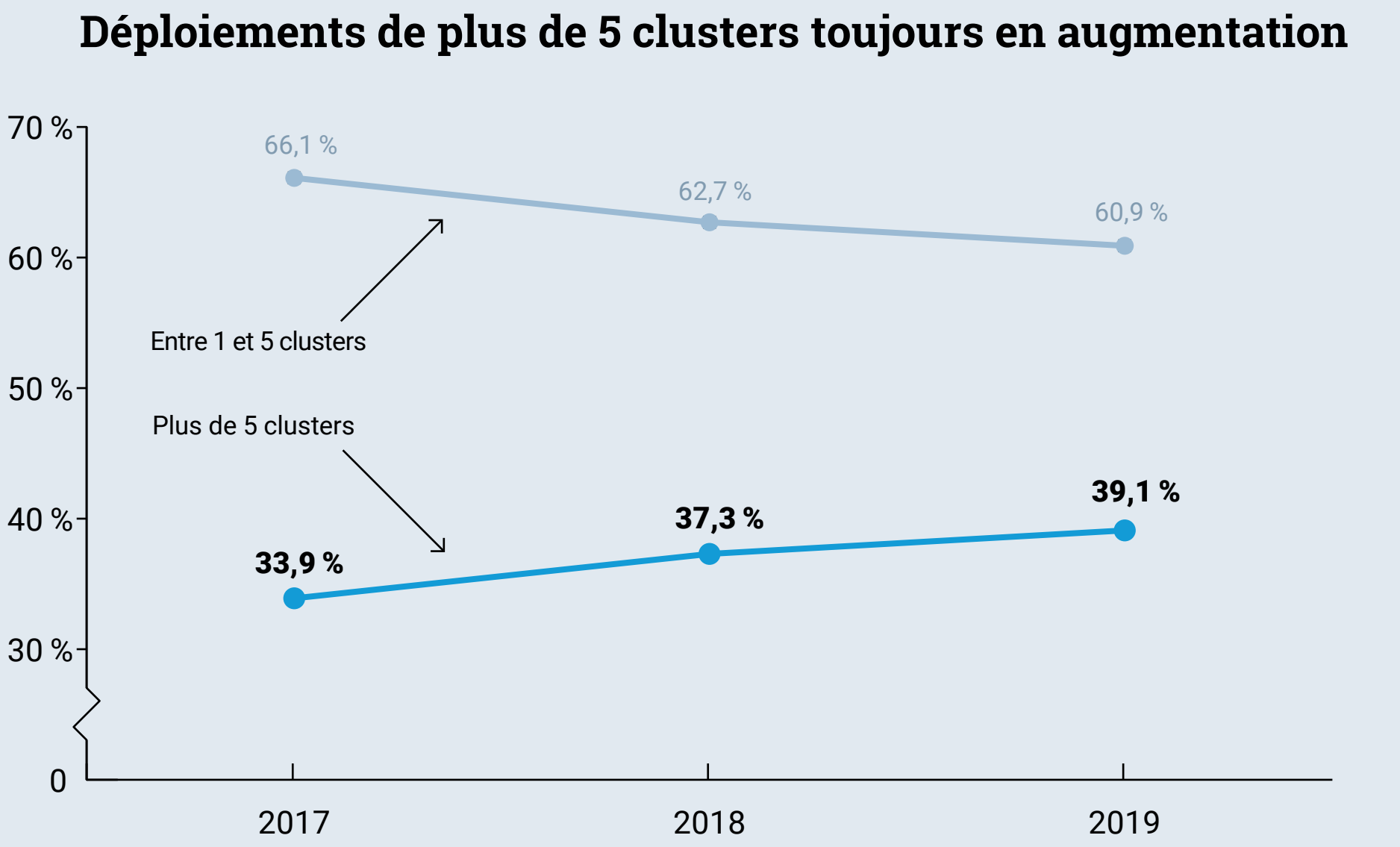
Bien que Kubernetes réponde aux enjeux de base des containers, comme la gestion du stockage, les utilisateurs restent confrontés à certaines problématiques notamment liées à la sécurité et au changement de culture. Ces derniers doivent aussi s'adapter aux innovations telles que le maillage des services : une technologie qui utilise des proxys side-car pour contrôler les microservices au sein des environnements Kubernetes, le but étant d'améliorer le contrôle du trafic, l'observabilité et la sécurité des applications pour les systèmes distribués. Penchons-nous un instant sur les données de l'enquête annuelle de la CNCF.

Les résultats de cette étude montrent bien l'adoption croissante de Kubernetes, qui est passée de l'expérimentation aux déploiements plus massifs. Ainsi, en 2017, 64 % des quelque 550 personnes interrogées affirmaient utiliser le projet Kubernetes en production. En 2018, la plateforme suscite déjà un plus vif intérêt, qui se traduit par plus de 2 400 participations au sondage. En 2019, 78 % des 1 300 répondants utilisaient Kubernetes en production, ce qui confirme sa place centrale au sein de la communauté cloud-native de la CNCF.

Mais quelle tendance a donné naissance à l'autre ? D'un côté, la démocratisation des containers a motivé la demande de solutions d'orchestration mais, de l'autre, on note que le déploiement de Kubernetes simplifie l'exécution de containers à grande échelle. En règle générale, les utilisateurs de Kubernetes exécutent un plus grand nombre de containers, mais leur réponse à la prolifération des containers est inégale.

Les déploiements Kubernetes de plus de cinq clusters continuent d'augmenter, passant de 34 % en 2017 à 39 % en 2019, comme le montre la figure 1.1 ci-dessous. D'ailleurs, plus de la moitié de ce groupe exécute au moins mille containers, une part qui progresse de 55 % à 62 % entre 2017 et 2019.

Figure 1.1 : *Les plus grands déploiements prédominent. Le pourcentage d'utilisateurs Kubernetes disposant de plus de cinq clusters a grimpé de 34 % à 39 % entre 2017 et 2019.*



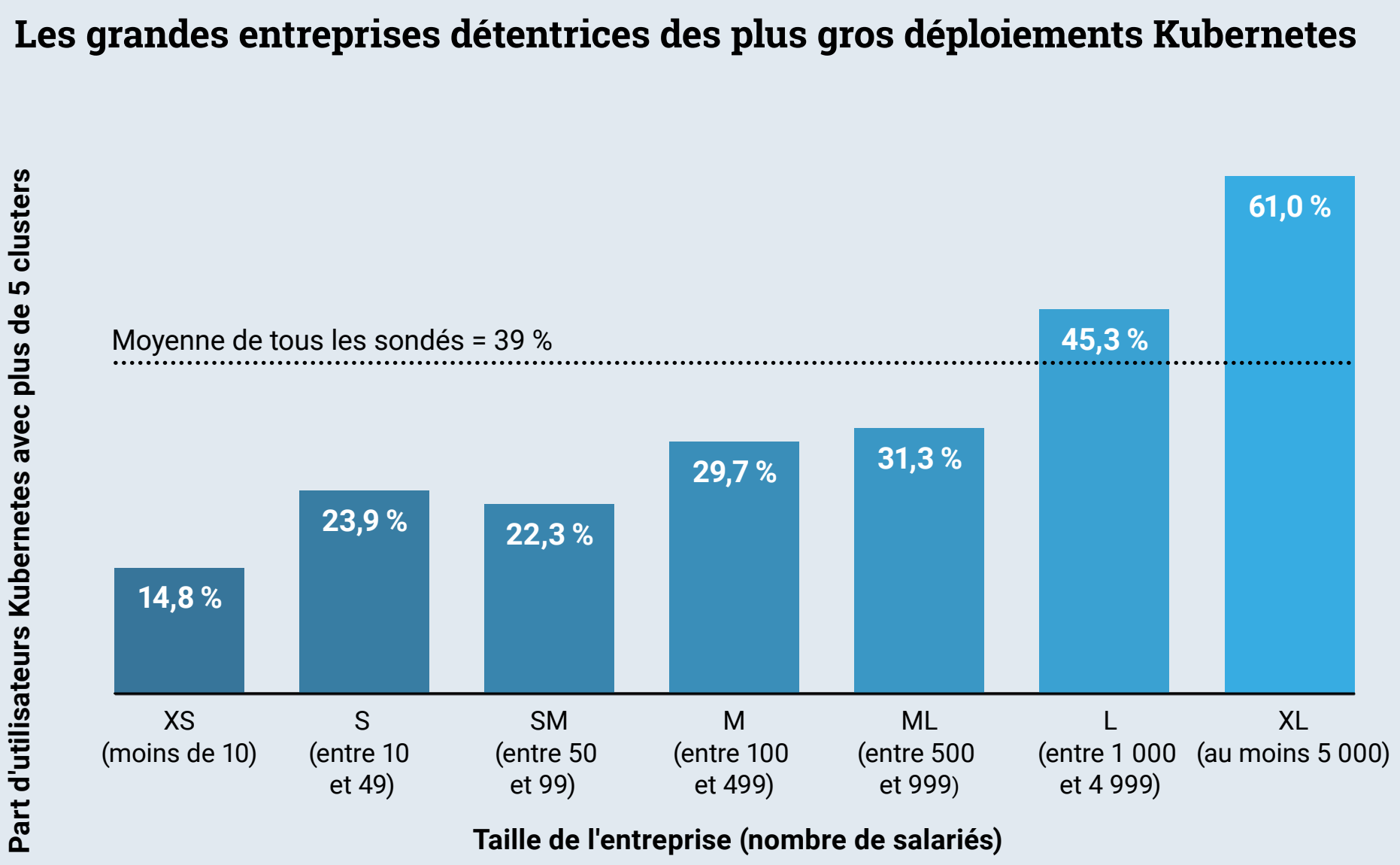
Source : analyse des enquêtes 2017, 2018 et 2019 de la CNCF par The New Stack.
Q. Quels sont les problèmes que soulèvent l'utilisation ou le déploiement de containers ?
Plusieurs réponses possibles. Ces données englobent uniquement les sondés avec au moins un cluster Kubernetes.
2017, n=454; 2018, n=1475; 2019, n=1197

À retenir :

- Les grandes entreprises sont les principales adeptes de Kubernetes, dont elles confient généralement le déploiement à leurs fournisseurs actuels.
- L'utilisation des containers est en hausse, mais les entreprises qui déploient cinq clusters Kubernetes ou moins éprouvent le plus de difficultés à s'adapter.
- Les utilisateurs ont recours à différentes solutions de stockage, d'accès et de maillage de services, qu'elles soient nouvelles ou déjà établies. Les outils émergents ont le plus de chances d'être testés pour répondre aux problématiques, mais il faut savoir que les clients ont tendance à conserver leurs fournisseurs actuels.

La figure 1.2 ci-dessous montre que l'effectif de l'entreprise va de pair avec l'envergure des déploiements Kubernetes, plus que le nombre de containers. 61 % des utilisateurs Kubernetes dans des entreprises d'au moins 5 000 salariés ont plus de cinq clusters.

Figure 1.2 : *61 % des utilisateurs Kubernetes d'entreprises comptant au moins 5 000 salariés ont plus de cinq clusters. L'étude rapporte une moyenne de 39 %.*



Source : analyse de l'enquête 2019 de la CNCF par The New Stack. Q. Si vous utilisez Kubernetes, combien de clusters exploitez-vous en production ? Moins de 10, n=81 ; entre 10 et 49, n=142 ; entre 50 et 99, n=121 ; entre 100 et 499, n=239 ; entre 500 et 999, n=80 ; entre 1 000 et 4 999, n=170 ; au moins 5 000, n=364.

L'étude révèle d'autre part que les entreprises d'au moins 1 000 salariés sont plus susceptibles d'avoir plus de cinq clusters, un chiffre en progression de 51 % à 56 % entre 2017 et 2019.

Ces données sont intéressantes, car elles montrent que les entreprises qui se distinguent par de larges déploiements Kubernetes observent une nette réduction des problèmes liés aux containers.

Les organisations qui déploient cinq clusters ou moins sont confrontées à des difficultés alors qu'elles s'efforcent de gérer un plus grand nombre de containers.

Orchestration : définition

Les containers et le DevOps sont deux tendances qui ont largement influencé l'infrastructure IT moderne. D'un côté, l'écosystème DevOps a évolué pour offrir une intégration, des tests, un déploiement et une surveillance continus, augmentant ainsi la vélocité du développement de logiciels. De l'autre, l'adoption des containers, associée aux bonnes pratiques reconnues du DevOps, permet d'assurer un déploiement rapide à grande échelle.

Mais bien que les containers contribuent en soi à la meilleure productivité des développeurs, les entreprises qui souhaitent aussi optimiser leurs processus DevOps et leurs investissements opérationnels auraient tort de se priver des nombreux avantages qu'apportent les outils d'orchestration :

- Gestion efficace des ressources
- Évolutivité transparente des services
- Haute disponibilité
- Frais généraux réduits à grande échelle
- Modèle déclaratif (pour la plupart des systèmes) visant à réduire les points de friction pour une gestion plus autonome
- Versant opérationnel de l'IaaS, associé à l'aspect gestion d'une PaaS (Platform as a Service)
- Expérience homogène entre les solutions sur site et dans le cloud public

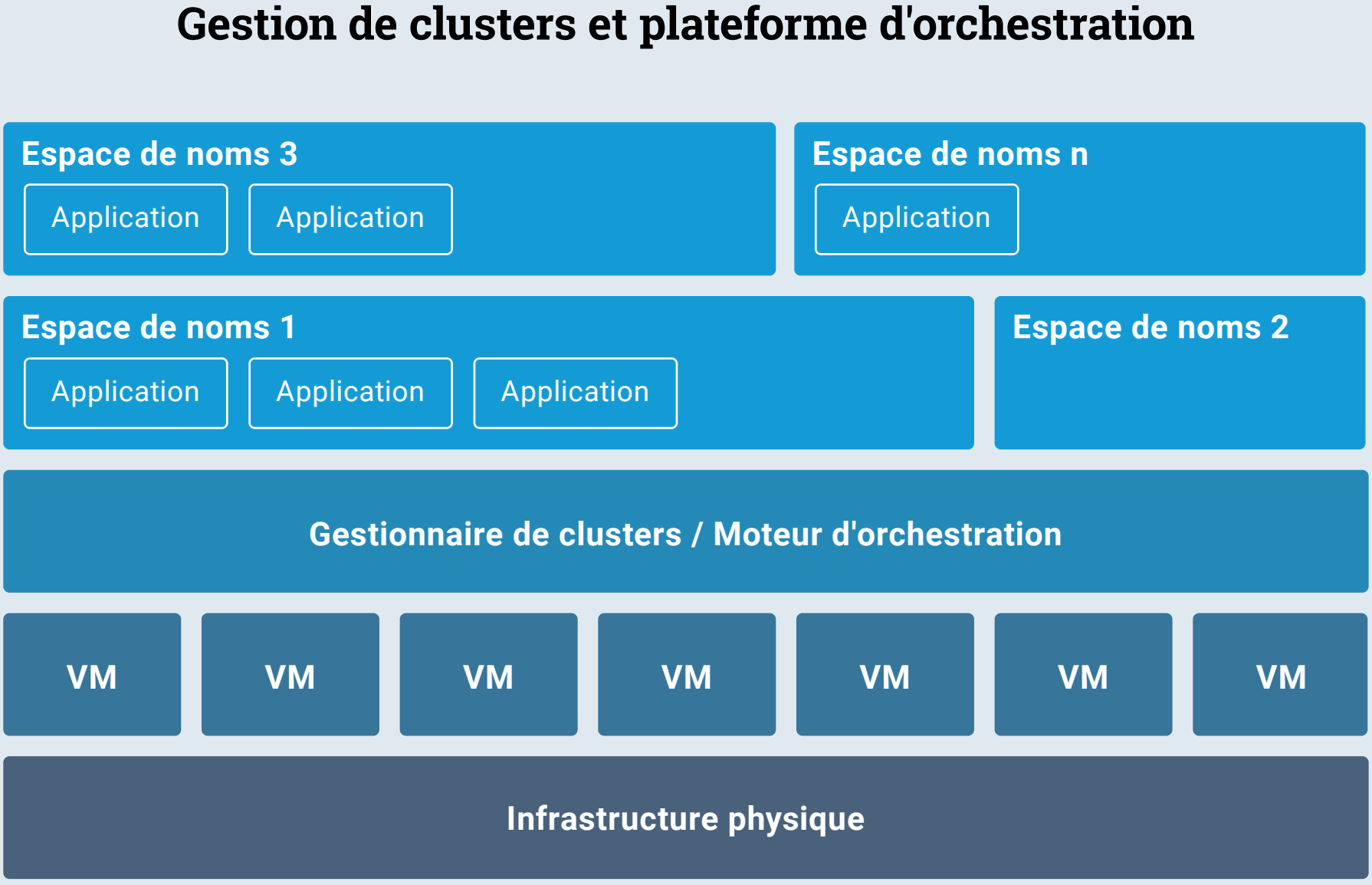


Figure 1.3 : *Les différentes couches de ressources d'un système, du point de vue du moteur d'orchestration des containers.*

Pour résumer, les opérateurs choisissent l'IaaS à des fins de contrôle et d'automatisation, tandis que les développeurs privilégient les PaaS pour la flexibilité, l'échelle et la productivité. Par conséquent, l'orchestration des containers offre le meilleur des deux mondes : l'automatisation et l'évolutivité.

Comme nous l'avons vu, l'introduction des containers a permis d'améliorer la productivité en développement, rendant le workflow DevOps parfaitement transparent : les développeurs pouvaient ainsi créer une image, exécuter un container et y écrire du code afin de le déployer dans des data centers locaux ou bien des environnements cloud publics. Pour autant, cette optimisation ne se traduit pas automatiquement par une meilleure efficacité en production.

En effet, l'environnement de production a peu à voir avec celui du développeur. Peu importe que l'on exécute une architecture à trois niveaux à grande échelle ou bien des applications basées sur les microservices, la difficulté consiste à gérer le grand nombre de containers ainsi que les nœuds

au sein du cluster. C'est précisément là qu'intervient l'orchestration et ses capacités d'automatisation.

La nature distribuée du cloud computing a profondément changé notre perception de l'infrastructure des machines virtuelles. La notion de « pet vs cattle » – on considère le container comme une unité de bétail interchangeable, à la différence d'un animal domestique – a contribué à façonner notre représentation des containers et de leur infrastructure.

En effet, ces derniers sont tous deux des éléments immuables, c'est-à-dire qu'ils ne sont plus jamais modifiés une fois déployés. Une mise à jour, un correctif ou une quelconque modification est nécessaire ? Alors de nouveaux containers ou serveurs, générés à partir d'une image commune, seront provisionnés en remplacement. Cette approche, que l'on pourrait comparer à la gestion du bétail, est à l'origine de l'analogie précédente.

De ce point de vue, les serveurs traditionnels et les machines virtuelles s'apparentent plutôt à nos bons vieux animaux domestiques : ni invariables, ni interchangeables, ils présentent un coût de maintenance élevé puisqu'ils nécessitent l'attention constante des équipes opérationnelles.

Les systèmes immuables, qui sont programmables, ouvrent donc la voie à l'automatisation. L'Infrastructure as Code (IaC) est au centre des solutions modernes : ce modèle permet aux applications de s'exécuter en programmant le provisionnement, la configuration et l'utilisation de l'infrastructure.

Ensemble, l'orchestration des containers, l'infrastructure immuable et l'automatisation pilotée par l'IaC répondent aux besoins de flexibilité et d'évolutivité.

En mettant cette notion en pratique, le déploiement et l'exploitation de containers à grande échelle ont élargi et enrichi les concepts d'évolutivité et de disponibilité des ressources.

Voici les fonctionnalités de base d'une plateforme d'orchestration des containers :

- Planification
- Gestion des ressources
- Découverte de services

- Contrôles d'intégrité
- Évolutivité automatique (autoscaling)
- Mises à jour/à niveau

Aujourd'hui, ce marché est dominé par Kubernetes : le système est à la fois reconnu par les entreprises, les fournisseurs de plateformes, les acteurs du cloud et les sociétés spécialisées dans l'infrastructure.

L'orchestration des containers favorise l'utilisation d'une architecture à microservices, où chaque application se compose de petits services indépendants et atomiques, chacun conçu pour une tâche unique. Les microservices sont empaquetés individuellement sous la forme d'un container, et ceux qui appartiennent logiquement à la même application sont orchestrés par Kubernetes à l'exécution.

L'essor de Kubernetes a entraîné dans son sillage l'apparition de nouveaux segments de marché basés sur la plateforme d'orchestration et de gestion des containers. Du stockage au réseau, en passant par la surveillance et la sécurité, de nouvelles entreprises et start-up développent des produits et services container-native. Dans le prochain chapitre, nous présenterons les composants de base des plateformes cloud-native ainsi que certaines start-up issues de cet écosystème émergent.

Architecture Kubernetes

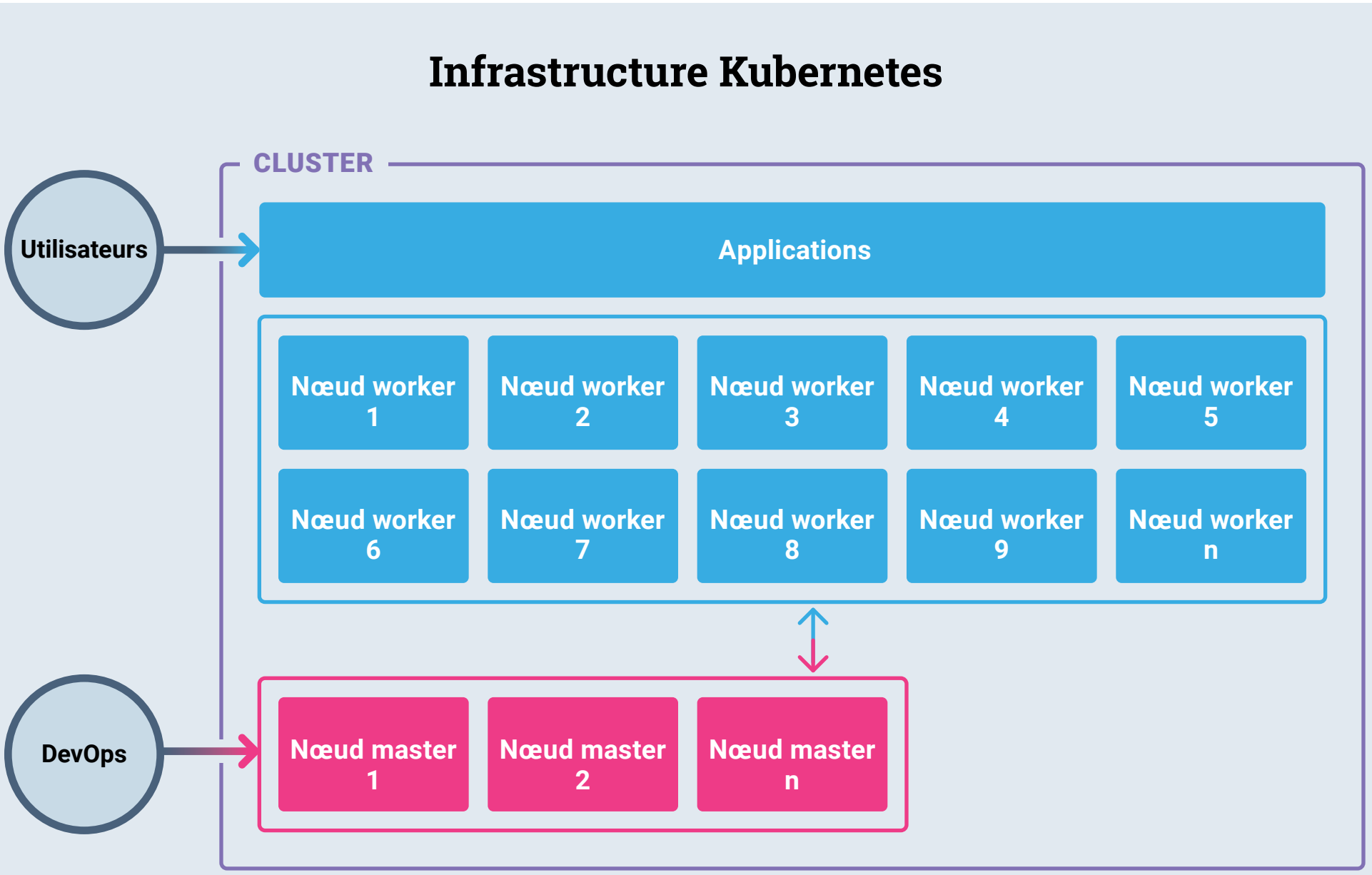
Une application contemporaine basée sur les containers et les microservices requiert une infrastructure solide afin de répondre aux besoins de clustering et d'orchestration dynamique. Celle-ci doit fournir les primitives de planification, de surveillance, de mise à niveau et de déplacement des containers entre les hôtes. Elle doit en outre gérer les primitives sous-jacentes de calcul, de stockage et de réseau en tant que pool de ressources. Chaque workload containerisé doit pouvoir exploiter les ressources auxquelles il est exposé, notamment les cœurs CPU, les unités de stockage et les réseaux.

Kubernetes (voir figure 1.4) est un système distribué open source qui utilise l'abstraction de l'infrastructure physique sous-jacente pour simplifier l'exécution d'applications containerisées à

l'échelle. Gérée du début à la fin par Kubernetes, une application se compose de containers qui sont regroupés et coordonnées sous la forme d'une seule unité. Une couche de gestion efficace du cluster permet à Kubernetes de découpler cette application de son infrastructure, comme illustré ci-dessous. Une fois que l'infrastructure Kubernetes est entièrement configurée, les équipes DevOps peuvent se concentrer sur la gestion des workloads déployés, sans devoir s'occuper du pool de ressources sous-jacent – CPU et mémoire – qui est géré par Kubernetes.

L'architecture de Kubernetes offre le parfait exemple d'un système distribué bien conçu. L'orchestrateur traite toutes les machines du cluster comme un pool de ressources unique. Il joue son rôle de système d'exploitation distribué en gérant efficacement la planification, l'allocation des ressources, la surveillance ainsi que le maintien de l'état souhaité de l'infrastructure et des workloads. Kubernetes est un système d'exploitation capable d'exécuter des applications modernes entre plusieurs clusters et infrastructures via les services cloud et dans les environnements de data centers privés.

Figure 1.4 : *Vision d'ensemble d'un cluster Kubernetes.*

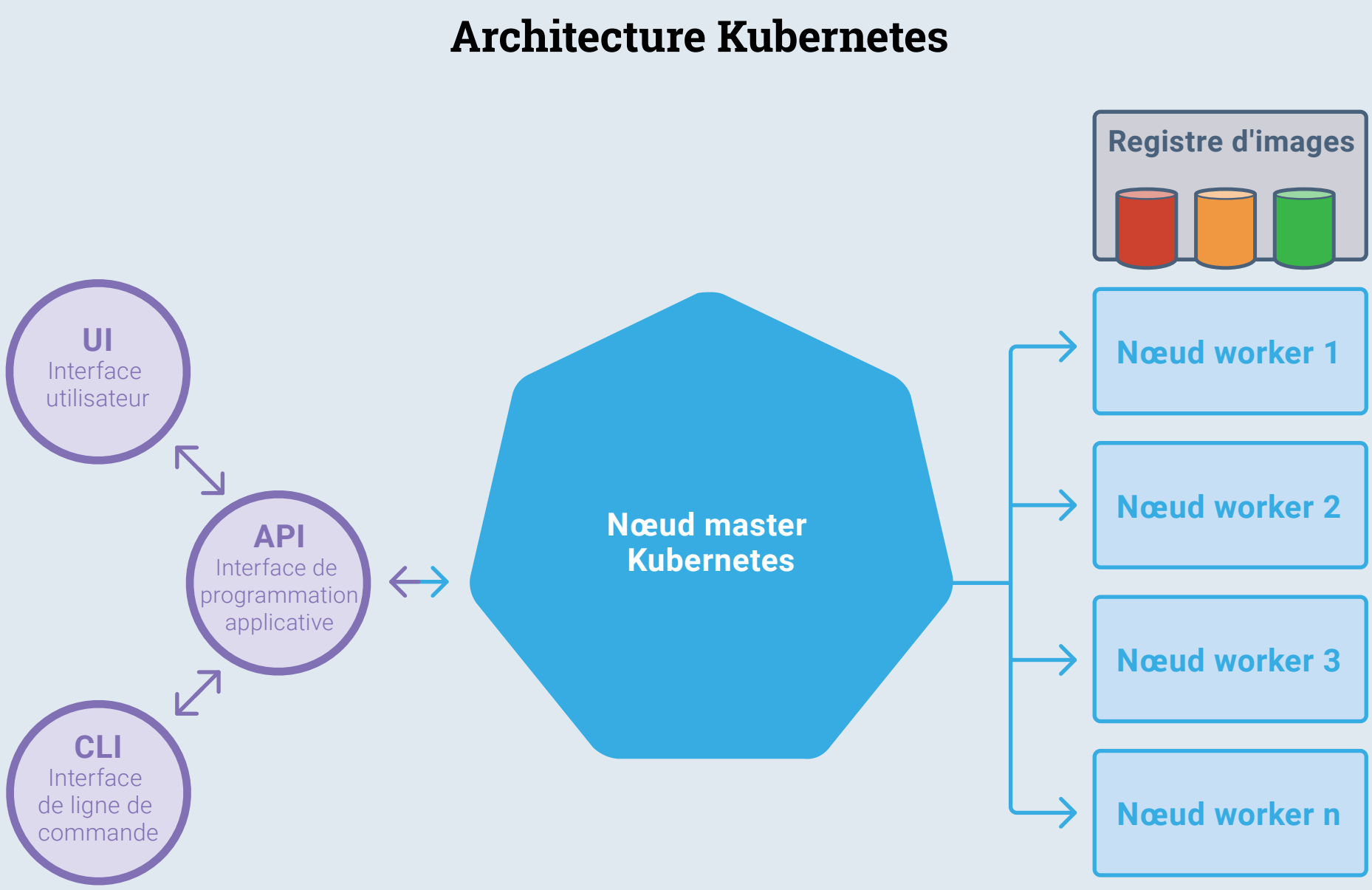


Source : Janakiram MSV

Comme tout système mature distribué, Kubernetes se compose de deux couches bâties sur le modèle maître-esclave : le nœud master et les nœuds worker (voir figure 1.5). Le master exécute le plan de contrôle qui est responsable de la planification et de la gestion du cycle de vie des workloads. Les nœuds worker, quant à eux, exécutent les applications. Ensemble, les nœuds master et worker forment un cluster.

Les équipes DevOps en charge du cluster communiquent avec l'API du plan de contrôle via l'interface de ligne de commande (CLI) ou bien des outils tiers. Les utilisateurs accèdent aux applications exécutées sur les nœuds worker. Enfin, les applications se composent d'une ou plusieurs images de containers qui sont stockées dans un registre accessible.

Figure 1.5 : *Le rôle du nœud master dans l'architecture Kubernetes.*



Source : Janakiram MSV

Plan de contrôle

Le plan de contrôle exécute les composants Kubernetes assurant les fonctionnalités de base : exposition de l'API Kubernetes, planification des déploiements de workloads, gestion du cluster et distribution des communications à travers l'ensemble du système. Comme l'illustre la figure 1.5, le master surveille les containers exécutés sur chaque nœud ainsi que l'intégrité de tous les nœuds enregistrés. Les images de containers, déployées en tant qu'artefacts, doivent être mises à la disposition du cluster Kubernetes via un registre privé ou public. Les nœuds responsables de la planification et de l'exécution des applications accèdent aux images de ce registre à partir du container runtime.

Comme le montre la figure 1.6, le master Kubernetes exécute les composants du plan de contrôle :

etcd

Développé par CoreOS, une société rachetée par Red Hat, etcd est un magasin de données clé-valeur distribué, léger et persistant, qui permet de maintenir les données de configuration du cluster. Il représente à tout moment l'état global du cluster, dont il constitue la référence unique. Les autres composants et services se réfèrent aux modifications du datastore etcd afin de maintenir l'état souhaité d'une application. Cette condition est définie par le biais d'une configuration déclarative qui énonce l'environnement optimal pour cette application que l'outil d'orchestration s'efforce alors d'atteindre. Cette configuration définit la manière dont l'orchestrateur doit aborder les propriétés d'une application, par exemple le nombre d'instances, les contraintes de stockage ou encore l'allocation des ressources.

La base de données etcd est seulement accessible via le serveur d'API. Par ailleurs, chaque composant du cluster passe par le serveur d'API pour lire ou écrire sur etcd, le cas échéant.

Serveur d'API

Le serveur d'API expose l'API Kubernetes au format JSON via HTTP, fournissant l'interface REST (Representational State Transfer) aux terminaux internes et externes de l'orchestrateur. La CLI, l'interface web ou d'autres outils peuvent passer une requête au serveur d'API. Le serveur traite puis valide la demande, avant de modifier l'état des objets d'API dans etcd. Ceci permet donc de configurer les workloads et les containers sur tous les nœuds worker.

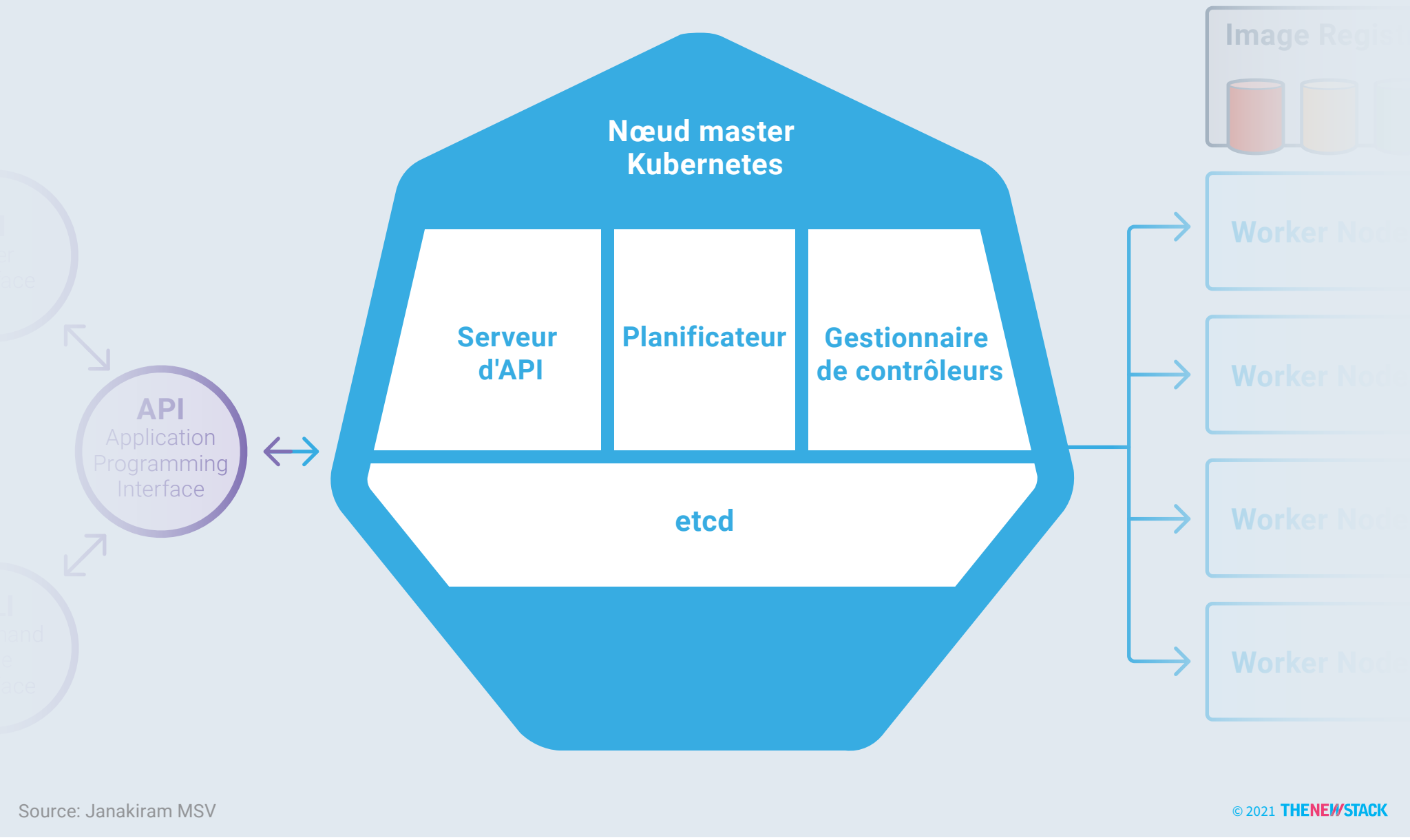


Figure 1.6 : *Composants du nœud master.*

Planificateur

Le planificateur choisit le nœud sur lequel doit s'exécuter chaque workload en fonction de l'estimation des ressources disponibles. Il surveille ensuite l'utilisation des ressources afin de garantir que le pod ne dépasse pas ses limites d'allocation. Ce composant du plan de contrôle maintient et mesure les besoins en ressources, leur disponibilité ainsi que diverses contraintes et directives politiques dictées par l'utilisateur, comme la qualité de service (QoS), les exigences d'affinité/anti-affinité et l'emplacement des données. Une équipe opérationnelle peut définir le modèle de ressources via une stratégie déclarative. Le planificateur interprète ces déclarations en tant que consignes de provisionnement et d'allocation de ressources pour chaque workload.

Gestionnaire de contrôleurs

L'architecture Kubernetes doit sa polyvalence à un autre élément du master : son gestionnaire de contrôleurs. À l'aide d'un contrôleur spécifique, son rôle est de garantir que le cluster maintient à tout moment l'état souhaité des applications. Dans ce contexte, un contrôleur désigne une boucle

de contrôle qui surveille l'état partagé du cluster via le serveur d'API et apporte des modifications en vue d'atteindre la condition désirée.

Le contrôleur assure la stabilité des nœuds et des pods en surveillant continuellement l'intégrité du cluster et des workloads qui y sont déployés. Par exemple, les pods exécutés sur un nœud malsain risquent de devenir inaccessibles. Dans pareil cas, la mission du contrôleur est de planifier le même nombre de nouveaux pods dans un nœud différent. Cette approche permet au cluster de maintenir l'état attendu à tout moment.

Kubernetes est fourni avec un ensemble de contrôleurs intégrés, exécutés à l'intérieur du gestionnaire de contrôleurs. Ceux-ci offrent des primitives alignées sur une certaine classe de workloads, comme les tâches avec ou sans état, les tâches cron planifiées, ou encore les tâches exécutées jusqu'à l'achèvement. Les développeurs et les opérateurs peuvent exploiter ces primitives tout en empaquetant et en déployant les applications dans Kubernetes.

Nœuds worker

Les nœuds worker sont en quelque sorte les muscles du cluster Kubernetes : ils exécutent les workloads containerisés et sont responsables des composants additionnels de journalisation, de surveillance et de découverte des services, ainsi que de modules optionnels. Leur rôle est d'exposer les ressources de serveurs, de réseaux et de stockage aux applications. Chaque worker inclut un runtime de containers, comme Docker, accompagné d'un agent (kubelet) qui communique avec le plan de contrôle. Un nœud peut correspondre à une machine virtuelle (VM) exécutée dans le cloud, ou bien à un serveur bare-metal au sein d'un data center.

Chaque nœud, comme le montre la figure 1.7, contient les éléments suivants :

Runtime des containers

Le runtime des containers, c'est-à-dire l'environnement d'exécution, gère le cycle de vie de chaque container du nœud. Lorsqu'un pod est programmé sur le nœud, le runtime extrait les images spécifiées par le pod à partir du registre. Dès qu'un pod est résilié, il élimine les containers qui lui sont associés. Kubernetes communique avec tous les environnements conformes aux normes Open Container Initiative (OCI), comme Docker et CRI-O.

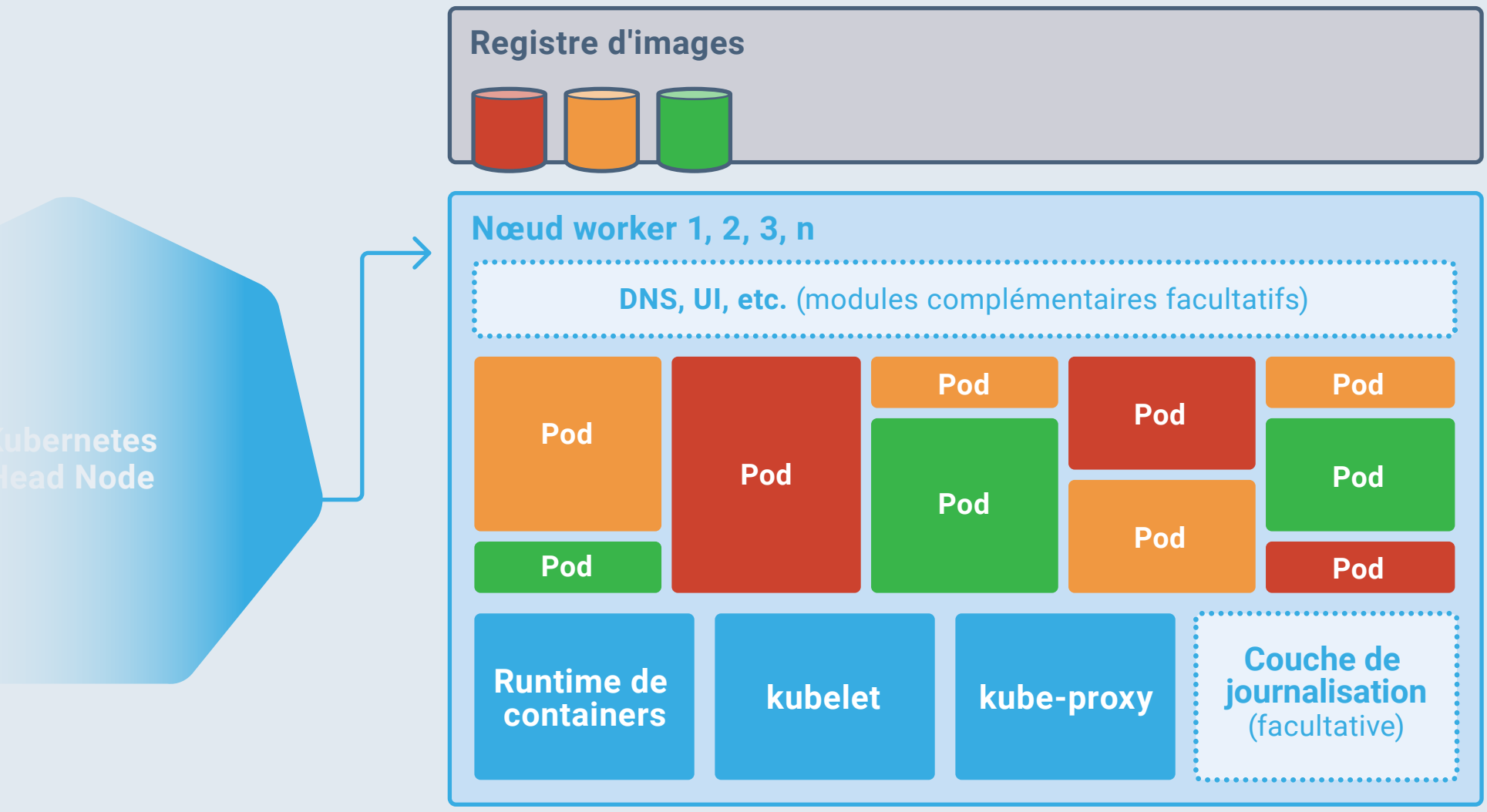


Figure 1.7 : *Composants des nœuds worker.*

L'OCI définit les spécifications des runtimes de containers et des formats d'images afin d'en promouvoir la normalisation.

kubelet

Le kubelet est l'agent Kubernetes chargé d'interagir avec le runtime de containers pour effectuer des opérations comme le démarrage, l'arrêt et le maintien des containers.

Chaque kubelet surveille également l'état des pods. Lorsque l'un d'entre eux ne correspond plus à l'état de référence tel que défini par le déploiement, il peut être redémarré sur le même nœud. L'état du nœud est transmis au plan de contrôle à quelques secondes d'intervalle via des messages de pulsation. Si le master détecte une anomalie, le contrôleur de réplication l'examine et planifie les pods sur d'autres nœuds sains.

kube-proxy

Le composant `kube-proxy` sert à la fois de proxy réseau et d'équilibreur de charge responsable

de l'orchestration du réseau : il achemine le trafic et les requêtes vers les pods adéquats en fonction du nom de service et du numéro de port associés à une demande entrante. Il exploite également les capacités réseau spécifiques aux OS en manipulant les politiques et les règles définies via `iptables`. Chaque `kube-proxy` peut s'intégrer aux couches réseau telles que Calico et Flannel.

Couche de journalisation

L'orchestrateur utilise fréquemment la journalisation afin de rassembler les métriques d'utilisation des ressources et de performance pour les containers de chaque nœud (CPU, mémoire, fichiers, réseau, etc.). La Cloud Native Computing Foundation utilise Fluentd : un composant logiciel offrant une couche de journalisation unifiée pour Kubernetes ou d'autres orchestrateurs. Celui-ci génère les métriques nécessaires au contrôleur principal de Kubernetes afin de mesurer la disponibilité des ressources du cluster ainsi que l'état de l'infrastructure globale.

Modules complémentaires

Kubernetes assure la prise en charge de services additionnels sous la forme de modules complémentaires. Ces services optionnels, comme le tableau de bord, sont déployés à l'image des autres applications mais s'intègrent à d'autres composants de base du nœud, comme la couche de journalisation et `kube-proxy`. Par exemple, le tableau de bord extrait les métriques du kubelet afin d'afficher une visualisation détaillée de l'utilisation des ressources. L'add-on DNS, basé sur `kube-dns` ou CoreDNS, optimise quant à lui `kube-proxy` via la résolution de noms.

Workloads

Si le plan de contrôle et les nœuds worker forment l'infrastructure centrale du cluster, les workloads correspondent quant à eux aux applications containerisées déployées dans Kubernetes.

Après avoir élaboré puis testé un microservice, les développeurs l'intègrent sous la forme d'un container, qui est la plus petite unité de déploiement rassemblée au sein d'un pod. Les ensembles de containers appartenant à la même application sont regroupés, empaquetés, déployés et gérés au sein de Kubernetes.

L'orchestrateur de containers expose les primitives à déployer tout en assurant constamment la mise à l'échelle, la découverte et la surveillance de ces microservices.

Les espaces de noms séparent les applications de façon logique et servent de cluster virtuel en définissant une limite et une portée pour toutes les ressources et les services d'une application donnée.

Les primitives Kubernetes suivantes sont déployées au sein d'un espace de noms :

Pods

Un pod est l'unité d'exécution basique d'une application Kubernetes, la plus simple et la plus petite dans le modèle d'objets de la plateforme. Il s'agit aussi du plus petit élément planifiable d'une application. Si Kubernetes est un système d'exploitation, alors un pod représente un ensemble de processus – chacun mappé sur un container – exécuté sur le cluster.

Le pod sert d'unité centrale de gestion des workloads pour Kubernetes : il représente une frontière logique pour les containers qui partagent un contexte et des ressources d'exécution. Cette capacité à regrouper les containers connexes compense l'un des défis de configuration introduits lorsque le modèle de la containerisation a supplanté la virtualisation de première génération, en permettant d'exécuter ensemble des processus dépendants.

Chaque pod rassemble donc un ou plusieurs containers qui s'échangent des informations via la communication entre processus (IPC) et peuvent partager la stack stockage et réseau. Ainsi, les containers qui doivent être couplés et colocalisés – par exemple un container de serveur web et un container de cache – peuvent être facilement empaquetés au sein d'un même pod. Aussi, un pod peut être mis à l'échelle soit manuellement, soit automatiquement, via la fonction d'autoscaling horizontal des pods (Horizontal Pod Autoscaling, HPA). Avec cette méthode, le nombre de pods faisant partie du déploiement augmente proportionnellement aux ressources disponibles.

Les pods offrent une séparation fonctionnelle entre développement et déploiement. Pendant que les développeurs peaufinent leur code, les opérateurs sont libres de décider quels containers connexes peuvent être regroupés dans une unité fonctionnelle. On obtient donc une portabilité

optimale, puisqu'un pod n'est qu'une collection d'une ou plusieurs images de containers gérées ensemble.

Contrôleurs

Dans Kubernetes, les contrôleurs enrichissent les pods de fonctionnalités additionnelles, comme la surveillance de l'état de configuration et différents critères d'exécution.

Par exemple, le contrôleur Deployment transmet des mises à jour déclaratives aux pods. Il garantit le maintien de l'état souhaité en surveillant l'intégrité des pods concernés. Chaque Deployment gère un ReplicaSet, dont le rôle est de maintenir le nombre de pods de réplicas exécutés dans un ensemble donné, en fonction de l'état souhaité.

Les Deployments arment les pods de capacités PaaS via des fonctions de mise à l'échelle, d'historique de déploiement et de retour arrière (rollback). Lorsqu'un Deployment est configuré avec un nombre minimal de deux réplicas, Kubernetes veille à ce qu'au moins deux pods soient toujours exécutés pour permettre la tolérance aux pannes. Même lorsque le pod est déployé avec un seul réplica, l'usage de la ressource Deployment reste fortement recommandé à la place de la spécification de base.

Le contrôleur StatefulSet est similaire au Deployment, mais il se destine plutôt aux pods qui requièrent un identificateur unique et persistant, ainsi qu'un ordre de création garanti. Dans les workloads tels que les clusters de base de données, StatefulSet créera un ensemble de pods hautement disponibles, dans un ordre donné, avec une convention d'affectation des noms prévisible. Les workloads avec état qui nécessitent un haut degré de disponibilité, comme Cassandra, Kafka, SQL Server et ZooKeeper, sont déployés sous la forme d'un objet StatefulSet dans Kubernetes.

D'autre part, le contrôleur DaemonSet permet de forcer l'exécution d'un pod sur chaque nœud d'un cluster. Kubernetes planifie automatiquement un DaemonSet dans chaque nouveau nœud worker provisionné, ce qui en fait le candidat idéal pour configurer et préparer les nœuds d'un workload. Par exemple, lorsqu'un partage de fichiers NFS (système de fichiers réseau) ou Gluster existant doit être monté sur les nœuds avant le déploiement du workload, il est conseillé d'empaqueter et de déployer le pod en tant que DaemonSet. Les agents de surveillance peuvent

aussi être utilisés en tant que DaemonSet, afin de garantir leur exécution dans chaque nœud.

Enfin, les pods associés aux tâches de traitement et de planification par lot peuvent être empaquetés en tant qu'objet Job ou CronJob. Un Job crée un ou plusieurs pods qu'il exécute jusqu'à ce qu'un nombre donné d'entre eux soit correctement résilié, ce qui signifie que la tâche est terminée. Un CronJob permet quant à lui d'exécuter une tâche planifiée au format crontab.

Les contrôleurs définissent le cycle de vie des pods en fonction des caractéristiques des workloads et de leur contexte d'exécution.

Services et découverte de services

Le modèle de services de Kubernetes offre à la fois l'aspect le plus basique et le plus important des microservices : la découverte.

Chaque objet d'API dans Kubernetes, y compris les nœuds et les pods, peut être associé à des paires clé-valeur : des métadonnées complémentaires permettant l'identification et le regroupement d'objets partageant un attribut ou une propriété en commun. Kubernetes désigne ces paires clé-valeur par les termes d'étiquettes et d'annotations. La découverte de services exploite les étiquettes et les sélecteurs pour associer un service à un ensemble de pods.

Un pod unique ou un ReplicaSet peut être exposé à des clients internes ou externes via les services, qui associent un ensemble de pods à un critère spécifique. Chaque pod dont l'étiquette correspond au sélecteur défini dans le manifeste du service sera automatiquement détecté par ce dernier. Cette architecture offre un mécanisme flexible à faible couplage pour la découverte de services.

Lorsqu'un pod est créé, l'adresse IP qui lui est attribuée est seulement accessible au sein du cluster. Mais il n'y a aucune garantie que cette adresse IP reste immuable tout au long du cycle de vie du pod. En effet, Kubernetes peut par exemple déplacer ou réinstancier les pods pendant la phase d'exécution, ce qui aboutit à la création d'une nouvelle adresse IP.

Pour compenser cette incertitude, les services veillent à ce que le trafic soit toujours acheminé

vers le pod approprié du cluster, quel que soit le nœud sur lequel il est planifié. Chaque service expose une adresse IP et potentiellement un nom DNS qui eux ne changeront jamais. Les consommateurs internes ou externes qui doivent communiquer avec un ensemble de pods utiliseront l'adresse IP du service ou bien son nom DNS. Le service fait ainsi office de ciment pour relier différents pods entre eux.

Le contrôleur Deployment s'appuie sur les étiquettes et les sélecteurs pour déterminer quels pods participeront à une opération de mise à l'échelle. Tout pod dont l'étiquette correspond au sélecteur défini par le service sera exposé via son terminal. Un service fournit ainsi un équilibrage de charge basique pour le routage du trafic entre pods correspondants.

Un sélecteur est un type de critère utilisé pour interroger les objets Kubernetes qui correspondent à une valeur d'étiquette. Cette technique efficace permet le couplage faible des objets. De nouveaux objets peuvent être générés dont les étiquettes correspondent à la valeur des sélecteurs. Les étiquettes et les sélecteurs constituent le mécanisme de regroupement principal de Kubernetes pour l'identification des composants auxquels s'applique une opération.

Pendant la phase d'exécution, les pods peuvent être mis à l'échelle via des ReplicaSets, garantissant que chaque Deployment exécute toujours le nombre voulu de pods. En effet, la ressource ReplicaSet maintient à tout moment un nombre prédéfini de pods. Lorsqu'un objet Deployment lance une mise à l'échelle, les nouveaux pods créés par cette opération commenceront immédiatement à recevoir du trafic.

Kubernetes propose trois méthodes d'exposition des services :

1. **ClusterIP :** permet aux pods de communiquer entre eux au sein du cluster.
   Par exemple, un pod de base de données exposé via un service ClusterIP devient disponible pour les pods de serveur web.
2. **NodePort :** permet d'exposer un service sur le même port pour tous les nœuds d'un cluster.
   Un mécanisme de routage interne veille à ce que la requête soit transmise au nœud approprié de chaque pod. NodePort est généralement utilisé pour les services avec consommateurs externes.

3. **LoadBalancer :** prolonge le service NodePort en ajoutant des équilibreurs de charge de couches L4 et L7. Cette méthode est souvent utilisée avec les clusters exécutés dans des environnements cloud publics prenant en charge le provisionnement automatique des équilibreurs de charge définis par logiciels.

Lorsque plusieurs services doivent partager le même équilibreur de charge ou un terminal externe, l'utilisation d'un contrôleur Ingress est recommandée. Celui-ci gère l'accès externe aux services d'un cluster – généralement HTTP – en fournissant un équilibrage de charge, une terminaison SSL (Secure Socket Layer) et un hébergement virtuel basé sur les noms.

L'Ingress est de plus en plus privilégié pour l'exécution de workloads de production dans Kubernetes. Cela permet aux différents microservices d'une application d'utiliser le même terminal, qui est exposé par un équilibreur de charge, une passerelle d'API ou un contrôleur de livraison d'applications (ADC).

Réseau et stockage

Le serveur, le réseau et le stockage constituent le socle des services d'infrastructure. Dans Kubernetes, les nœuds représentent la composante calcul (serveur), qui fournit les ressources de base aux pods exécutés dans les clusters. Les services de réseau et de stockage sont fournis par des plug-ins container-native définis par logiciels, spécialement conçus pour Kubernetes.

La composante réseau assure la communication de pod à pod, de nœud à pod, de pod à service ainsi qu'entre les clients externes et les services. Kubernetes suit un modèle d'implémentation réseau basé sur les plug-ins. Kubenet, qui est le plug-in réseau par défaut, est simple à configurer. Il s'utilise généralement avec un fournisseur cloud qui définit les règles de routage pour la communication entre les nœuds, ou bien dans des environnements à simple nœud.

Kubernetes assure la prise en charge de plusieurs plug-ins basés sur la spécification Container Network Interface (CNI), qui définit la connectivité réseau des containers et gère les ressources réseau lorsque le container est supprimé. Il existe plusieurs implémentations de CNI, comme Calico, Cil3ium, Contiv, ou encore Weave Net. La spécification CNI prend également en

charge les réseaux virtuels disponibles dans le cloud public, ce qui permet d'étendre la topologie réseau et les sous-réseaux aux clusters Kubernetes.

Certains plug-ins réseau conformes CNI, comme Calico, implémentent des politiques de routage strictes via des pods isolés. Ils introduisent des règles de pare-feu aux pods et aux espaces de noms d'un cluster Kubernetes.

Le stockage persistant est exposé à Kubernetes via des volumes persistants. Les pods consomment les volumes via des revendications de volumes persistants. Les administrateurs provisionnent le stockage en créant les volumes persistants à partir des périphériques de stockage NAS, SAN, DAS, SSD, NVMe ou flash. Les développeurs et les équipes DevOps obtiennent une partie des volumes persistants en revendiquant ceux qui sont associés aux pods.

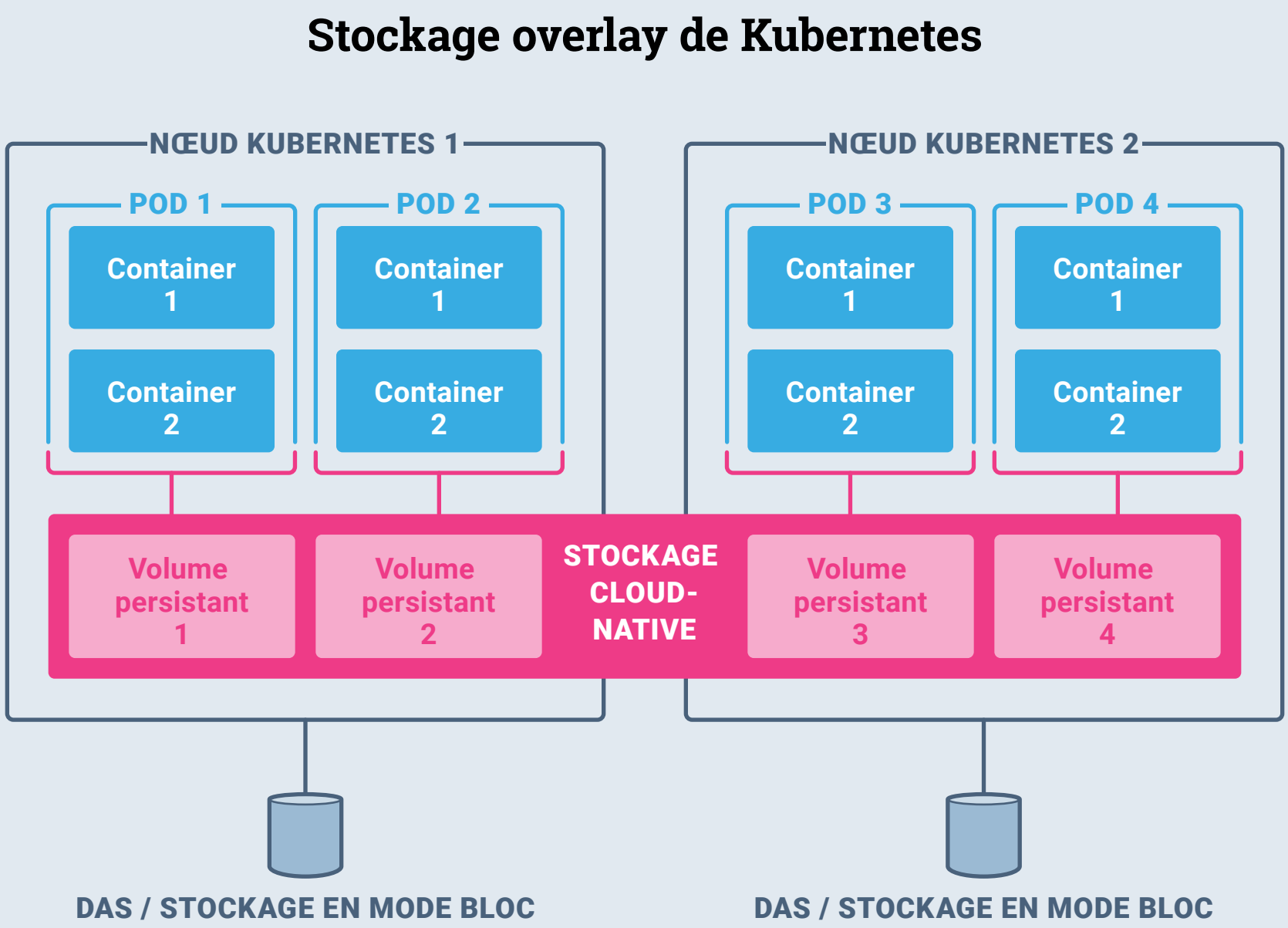
Kubernetes est fourni avec des primitives de stockage qui exposent les volumes issus des nœuds existants. Ces primitives sont un type de volume dont le rôle est de rendre le stockage sous-jacent accessible aux pods. Ces types de volumes comprennent par exemple `emptyDir` et `hostPath`, chacun utilisé dans des cas précis : `emptyDir` est un volume vierge, tandis que `hostPath` sert à rendre les volumes locaux disponibles pour les pods. Mais ces derniers se caractérisent par un manque de disponibilité et de tolérance aux pannes, en raison de leur couplage fort avec le nœud. Les couches de stockage overlay (voir figure 1.8) rassemblent les volumes à partir du stockage de blocs, des NAS et des SAN pour exposer le stockage externe aux objets Kubernetes.

Pour garantir une haute disponibilité et offrir des capacités de stockage container-native, Kubernetes a introduit des plug-ins permettant aux fournisseurs d'exposer leurs plateformes aux workloads containerisés. Le stockage de blocs des fournisseurs de cloud public, les systèmes de fichiers distribués basés sur NFS et GlusterFS, ainsi que certaines plateformes de stockage commerciales disposent ainsi de plug-ins inclus dans la distribution de base open source de Kubernetes. Les administrateurs créent des classes de stockage en fonction des performances et du débit de chaque type de moteur de stockage. Des volumes persistants et des revendications peuvent être créés à partir de ces classes de stockage pour différents types de workloads.

Par exemple, un système de gestion de base de données relationnelle (RDBMS) peut être associé à une classe de stockage caractérisée par un grand nombre d'opérations d'entrée/sortie par seconde (IOPS), alors qu'un système de gestion de contenu (CMS) ciblera un moteur de stockage distribué via une autre classe.

À l'instar des CNI, la communauté Kubernetes a défini des spécifications de stockage via les CSI (Container Storage Interface). Cette méthode favorise une approche standard et portable de l'implémentation et de la consommation de services de stockage via les workloads containerisés.

Figure 1.8 : *Exposition du stockage aux pods et aux containers.*



Source : Janakiram MSV

Kubernetes, clé de voûte de l'informatique web-scale

Kubernetes, en digne héritier de Borg, est conçu pour les workloads à très grande échelle. Son architecture moderne permet une exploitation optimale des ressources d'infrastructure. L'ajout de nœuds worker à un cluster existant se fait en toute simplicité, presque sans avoir à modifier sa configuration. Les workloads tirent ainsi immédiatement parti des ressources CPU, de la mémoire et du stockage des nouveaux nœuds.

Le regroupement de containers connexes au sein d'un pod, qui est traité comme une unité de déploiement et d'échelle, permet d'améliorer les performances. Par exemple, les containers de serveur web et de cache colocalisés au sein d'un même pod offrent une latence réduite. Les containers du pod partagent le même contexte d'exécution et peuvent donc utiliser la communication entre processus pour diminuer les frais généraux.

D'autre part, les pods appartenant aux mêmes ressources ReplicaSet et Deployment offrent une mise à l'échelle accélérée. Quelques secondes suffisent en effet pour dimensionner un déploiement de centaines de pods. Ces derniers sont planifiés sur les nœuds en fonction des ressources disponibles et de l'état de configuration souhaité. Grâce à la fonction d'autoscaling horizontal des pods (HPA), Kubernetes peut monter ou baisser automatiquement les capacités d'un déploiement.

Lorsqu'il est exécuté dans un environnement d'infrastructure élastique, l'orchestrateur peut utiliser Cluster Autoscaler pour ajouter et supprimer des nœuds du cluster. Conjuguée au HPA, cette technique permet de gérer efficacement l'autoscaling dynamique du workload et de l'infrastructure.

La stack réseau légère et la découverte de services de Kubernetes sont taillées pour une application à grande échelle. Elles peuvent en effet gérer des dizaines de milliers de terminaux exposés par les services pour une consommation interne et externe.

L'écosystème et la communauté Kubernetes innovent sans cesse afin d'adapter la plateforme aux workloads à très grande échelle.

Adopter Kubernetes

Kubernetes est déployé dans les environnements de production en tant que moteur d'orchestration des containers, PaaS et infrastructure centrale pour la gestion d'applications cloud-native. Ces cas d'usage ne sont pas mutuellement exclusifs : les opérateurs peuvent tout à fait déléguer la gestion complète du cycle de vie des applications (ALM) à une couche PaaS basée sur Kubernetes. Ils peuvent également utiliser un déploiement Kubernetes autonome pour gérer les applications déployées à l'aide de la chaîne d'outils CI/CD existante. Aussi, les développeurs greenfield peuvent exploiter Kubernetes pour gérer les nouvelles applications cloud-native basées sur les microservices, notamment via des scénarios avancés comme les mises à niveau progressives et les déploiements de type canary.

Les clients qui testent Kubernetes ou l'utilisent pour des déploiements de production peuvent choisir parmi une large sélection de distributions alternatives.

Voici une liste des principales solutions Kubernetes disponibles sur le marché.

Distribution de base

La distribution open source de base – « upstream » – de Kubernetes, disponible dans GitHub, peut être déployée dans les data centers, ainsi que dans les environnements cloud publics et privés. Le codebase comprend les composants centraux et les éléments requis pour exécuter des workloads sur Kubernetes.

La solution est fournie avec un outil de déploiement nommé kubeadm, qui simplifie la configuration de clusters basés sur CentOS, Red Hat Enterprise Linux, SUSE, Ubuntu et d'autres distributions Linux.

Les clients peuvent aussi utiliser des outils de déploiement automatiques comme kops, kubespray et Rancher Kubernetes Engine pour créer et configurer les clusters. Ces solutions offrent un degré de personnalisation plus ou moins avancé pour la configuration des runtimes de containers ainsi que des plug-ins réseau et de stockage.

La distribution de base est gratuite, transparente et offre une approche entièrement

personnalisable de l'installation de Kubernetes. Toutefois, elle demande un certain niveau d'expertise et de compétences.

Distributions commerciales

Certains fournisseurs proposent une version personnalisée et optimisée de Kubernetes, introduisant des services et un support professionnels. Cette approche s'inscrit dans le modèle traditionnel de commercialisation de logiciels open source via les services.

Les distributions commerciales accélèrent la configuration et le déploiement des clusters Kubernetes, tout en promettant des correctifs, des hotfix et des mises à niveau transparentes vers les nouvelles versions de la plateforme.

Parmi ces fournisseurs, nous pouvons citer Canonical, D2iQ, HPE, Mirantis, Rancher ou encore VMware.

Containers as a Service

Kubernetes est aussi disponible sous la forme d'une plateforme entièrement managée et hébergée. Data center d'entreprise, infrastructure en colocation, cloud public… quel que soit l'environnement, les fournisseurs de Containers as a Service (CaaS) promettent une solution Kubernetes intégrale, soutenue par un accord de niveau de service (SLA) commercial.

Aujourd'hui, presque tous les grands acteurs du cloud proposent une offre CaaS. Voici quelques exemples de CaaS dans le cloud public : Alibaba Container Service for Kubernetes, Amazon EKS, Azure AKS, DigitalOcean Kubernetes, Google Kubernetes Engine, Huawei Cloud Container Engine et IBM Kubernetes Service.

Mirantis, NetApp et Platform 9 offrent quant à eux une solution CaaS pour les data centers et les environnements cloud privés.

Platform as a Service

Souvent, les clients déploient une plateforme PaaS pour normaliser leurs environnements

de développement et de déploiement. En utilisant une PaaS basée sur Kubernetes, ils pourront développer à la fois des applications métiers traditionnelles et entièrement nouvelles (greenfield). Les fournisseurs classiques ont adopté Kubernetes afin d'offrir aux entreprises une solution PaaS de bout en bout.

Les offres basées sur Kubernetes s'appuient sur ses capacités d'orchestration de containers pour proposer une gestion complète du cycle de vie des applications containerisées.

Red Hat OpenShift et VMware Tanzu Kubernetes Grid sont des exemples de solutions PaaS basées sur Kubernetes.

Kubernetes, plan de contrôle universel

Kubernetes s'impose comme l'un des meilleurs plans de contrôle des applications et des infrastructures de nouvelle génération. Son puissant planificateur, initialement conçu pour gérer le placement approprié des pods sur les nœuds, est assez extensible. Il permet de résoudre la plupart des problèmes existant au sein des systèmes distribués traditionnels.

Kubernetes est bien parti pour devenir le plan de contrôle de référence des tâches de planification et de gestion au sein d'environnements hautement distribués. Celles-ci comprennent par exemple le déploiement de machines virtuelles sur des hôtes physiques, l'installation de containers sur les équipements de périphérie, ou encore l'extension du plan de contrôle à d'autres planificateurs, comme les environnements sans serveur.

Des serveurs bare-metal aux machines virtuelles, en passant par les objets connectés (IoT) et les services managés dans le cloud, Kubernetes a dépassé le concept de container et de pod pour répondre à de multiples enjeux de provisionnement et de planification.

Les quelques exemples qui suivent soulignent cette tendance.

Crossplane

Crossplane entend normaliser la gestion de l'infrastructure et des applications par le biais d'une approche basée sur l'automatisation, la configuration déclarative et les API, en suivant la voie

tracée par Kubernetes. Ce plan de contrôle unifié s'intègre parfaitement aux outils et systèmes existants, tout en simplifiant l'élaboration de politiques, de quotas et de rapports de suivi.

Crossplane agit donc comme une interface entre Kubernetes et les workloads traditionnels tels que les bases de données, et même les services managés dans le cloud public. Les équipes DevOps peuvent déclarer des ressources externes via la même spécification YAML, au côté des applications natives de Kubernetes. Cette approche encourage la configuration en tant que code en étendant le contrôle de version, l'intégration et le déploiement continus aux ressources extérieures à Kubernetes.

K3s

K3s, de Rancher, est une distribution Kubernetes certifiée, dédiée aux workloads de production qui sont exécutés dans des environnements soumis à de fortes contraintes, comme les déploiements IoT et Edge Computing.

K3s peut être déployé sur les machines les plus virtuelles du cloud public, ou même sur un Raspberry Pi. Son architecture, qui reste entièrement compatible et conforme aux tests Kubernetes de la CNCF, est spécialement optimisée pour les déploiements sans surveillance et distants sur des équipements aux ressources limitées.

En le rendant à la fois accessible et léger, K3s introduit Kubernetes dans la couche de l'Edge Computing.

KubeEdge

Lors de la KubeCon+CloudNativeCon 2018 de Seattle, Huawei avait présenté KubeEdge, le projet officiel visant à exploiter la puissance de Kubernetes en périphérie.

KubeEdge repose sur l'Intelligent Edge Fabric (IEF) : une plateforme d'IoT Edge basée sur la PaaS IoT du géant chinois. Une grande partie de l'IEF a été modifiée et transformée en open source pour donner KubeEdge. Disponible dans une version 1.3, KubeEdge est stable et répond aux principaux cas d'usage liés à l'IoT ainsi qu'à la périphérie. Il peut être installé sur une distribution Linux prise en charge et sur un appareil ARM, par exemple un Raspberry Pi.

Le projet KubeEdge fait partie de la CNCF Sandbox.

KubeVirt

KubeVirt, un module complémentaire de gestion de machines virtuelles pour Kubernetes, permet aux utilisateurs d'exécuter des VM au côté de containers dans leurs clusters Kubernetes ou OpenShift. L'add-on étend les fonctionnalités de Kubernetes en ajoutant des types de ressources pour les VM et les ensembles de VM via l'API CustomResourceDefinitions (CRD) de l'orchestrateur. Les VM KubeVirt s'exécutent au sein de pods Kubernetes, où elles accèdent aux ressources de stockage et de réseau standard et peuvent être gérées à l'aide d'outils Kubernetes classiques comme `kubectl`.

KubeVirt fait également partie des projets CNCF Sandbox.

Virtual Kubelet

Le projet Virtual Kubelet de Microsoft représente l'extension la plus intéressante de l'agent kubelet et de l'API Kubernetes. Le Virtual Kubelet est un agent exécuté dans un environnement externe, enregistré en tant que nœud du cluster Kubernetes. Il crée une ressource de nœud via l'API Kubernetes. En exploitant l'approche des teintes et des tolérances, l'agent planifie les pods dans un environnement externe en appelant son API native.

Virtual Kubelet fonctionne avec Azure Container Instances, Azure IoT Edge et le plan de contrôle AWS Fargate.

Après des débuts modestes dans le monde de l'orchestration de containers, l'évolution de Kubernetes lui a permis de s'imposer rapidement en tant que système d'exploitation taillé pour le cloud et la périphérie. Kubernetes est le socle de l'infrastructure moderne des data centers, du cloud hybride, du cloud public et des environnements multicloud.

Dans le prochain chapitre, nous explorerons l'écosystème croissant de Kubernetes et l'industrie cloud-native.

Architectures en self-service et opérateur Kubernetes pour Cassandra

Ces dix dernières années ont été marquées par l'essor des bases de données distribuées (base de données open source Apache Cassandra, par exemple) conçues pour compléter Kubernetes et son architecture évolutive. Cette tendance a inauguré l'ère des architectures entièrement fournies en self-service, un sujet que nous abordons dans ce podcast en compagnie de Kathryn Erickson et de Patrick McFadin, deux collaborateurs de DataStax, une entreprise spécialisée dans les plateformes NoSQL cloud-native.

DataStax fournit un guide pour Cassandra via un nouvel opérateur Kubernetes destiné à abstraire les couches de la base de données pour permettre aux développeurs de se recentrer sur leur cœur de métier : coder.

Les architectures en self-service donnent aux développeurs la possibilité d'innover dans leur manière d'utiliser les données. Ou comme le dit Patrick McFadin, « les bases de données [sont] moins un fardeau cognitif pour les développeurs qu'elles ne l'étaient auparavant. »

Écouter sur SoundCloud **Visionner sur YouTube**

Kathryn Erickson *dirige la stratégie des solutions en self-service pour DataStax.*

Patrick McFadin *est évangéliste en chef d'Apache Cassandra et vice-président des relations avec les développeurs chez DataStax.*

L'observabilité de l'IA pour réduire la complexité de Kubernetes

Avec Kubernetes, faire évoluer des applications sur plusieurs environnements cloud est devenu une réalité. Mais cela a également engendré beaucoup de complexité au sein des départements informatiques.

Activiste DevOps pour la plateforme d'intelligence logicielle Dynatrace, Andreas Grabner a récemment constaté que les environnements Kubernetes d'entreprise impliquaient de « prendre en compte des milliards d'interdépendances ». Oui, des milliards.

Dans ce podcast, il explique comment l'IA permet de maîtriser cette complexité de Kubernetes. Nous revenons également sur des cas d'usage de l'observabilité de l'IA pour les opérateurs et les développeurs, l'intérêt des données de télémétrie pour les équipes de gestion de Kubernetes, et les changements culturels dans les équipes de développement à l'ère de Kubernetes.

Écouter sur SoundCloud **Visionner sur YouTube**

Andreas Grabner *est activiste DevOps chez Dynatrace. Il aide les développeurs, les testeurs, les opérateurs et les professionnels xOps à exploiter les solutions Dynatrace pour gagner en efficacité.*

Comment adapter des applications data-centric aux architectures Kubernetes

Comme les microservices, Kubernetes est un système « faiblement couplé » par nature. Les composants sont simples, ont un seul objectif et fonctionnent de manière indépendante. En revanche, un code intégré interdépendant peut être plus difficile à mettre à jour lorsque des modifications sont apportées à l'application.

Technologue émérite et Lead Data Scientist chez HPE, Nanda Vijaydev explique comment la gestion d'applications data-centric dans ce contexte de faible couplage exige une attention particulière à bien des niveaux.

« Comment exploiter dynamiquement un système conçu comme modèle de données… et le déployer à grande échelle ? » s'interroge-t-elle. « C'est là que la technologie évolue. »

Grâce à ses diverses acquisitions (MapR Technologies, BlueData, Cloud Technology Partners (CTP), etc.), HPE renforce sa capacité à aider les entreprises à relever les défis associés au couplage des workloads data-centric exécutés sur Kubernetes. Sa plateforme logicielle permet aux utilisateurs d'envisager la gestion des données non plus comme un problème Kubernetes, mais plutôt comme un problème de stockage ou un problème de réseau.

Écouter sur SoundCloud

Nanda Vijaydev est une technologue émérite et Lead Data Scientist chez HPE, où elle exploite des technologies comme Kubernetes, TensorFlow, H2O et Spark pour créer des solutions de machine learning et de deep learning pour les entreprises.

CHAPITRE 2

Mapper l'écosystème Kubernetes

Kubernetes a gagné la bataille des outils d'orchestration pour devenir la plateforme de gestion de containers privilégiée. Ce système s'est imposé comme le logiciel d'infrastructure unifiée de référence pour les data centers d'entreprise, les plateformes de cloud public, les plateformes de cloud hybrides, les appliances d'infrastructure hyperconvergées et même l'Edge Computing.

Socle d'une infrastructure moderne, Kubernetes exécute les workloads actuels, souvent dénommés applications cloud-native. La couche d'infrastructure principale est renforcée par des projets open source et des logiciels commerciaux constituant l'environnement cloud-native.

Ce chapitre vous offre un panorama complet sur les technologies cloud-native, ainsi qu'une exploration détaillée de leurs composantes clés.

Du système d'exploitation à l'expérience des développeurs, nous passerons au crible cet environnement afin d'examiner de plus près les principaux projets open source et offres commerciales qui proposent des technologies cloud-native aux entreprises et aux utilisateurs.

Veuillez noter que ce chapitre se limitera à la technologie Kubernetes. Par exemple, les systèmes de bases de données font partie de l'écosystème global Kubernetes. Les DBaaS (Databases as a Service) en particulier ont gagné en importance au cours des dernières années. Elles intègrent aujourd'hui des produits tels qu'Amazon Aurora, Azure SQL Database, MongoDB Atlas et Redis Cloud Essentials. Les bases de données ne sont toutefois pas une composante clé des

environnements cloud-native, mais plutôt un workload. Élément majeur des bases de données et des workloads avec état, le stockage joue un rôle déterminant dans l'environnement cloud-native que nous étudierons plus bas.

L'essor du cloud-native et du CaaS

Selon la fondation Linux, les technologies cloud-native s'appuient sur un environnement logiciel open source pour déployer des applications telles que les microservices. Pour ce faire, elles packagent chaque partie dans leurs propres containers et orchestrent dynamiquement ces containers pour optimiser l'utilisation des ressources.

« Cloud-native » est une expression utilisée pour décrire les environnements basés sur les containers. Les technologies cloud-native sont utilisées pour développer les applications intégrant des services packagés dans des containers, déployées en tant que microservices et managées sur des infrastructures élastiques par le biais de processus DevOps agiles et de workflows de livraison continus.

Caractéristique clé des technologies cloud-native, la portabilité nécessite d'utiliser une infrastructure homogène sur tous les environnements. Dans cette optique, Kubernetes devient le plus petit dénominateur commun de l'infrastructure, mais aussi le socle de l'environnement cloud-native.

Bien que Kubernetes soit un élément essentiel de cet environnement, les développeurs et les ingénieurs DevOps ont également besoin de logiciels supplémentaires pour déployer, adapter et gérer les applications modernes. Les fournisseurs tels que Red Hat et VMware proposent des plateformes de bout en bout basées sur Kubernetes. Les fournisseurs de cloud public – y compris Amazon Web Services (AWS), Google Cloud Platform (GCP) et Microsoft Azure — offrent des services managés basés sur Kubernetes fonctionnant sur des infrastructures informatiques, de stockage et réseau existantes.

Les plateformes de gestion de containers intégrées basées sur Kubernetes ont créé une nouvelle catégorie de modèle de déploiement applicatif : le Containers as a Service (CaaS). Au même titre qu'une PaaS (Platform as a Service), la plateforme de gestion de containers peut être déployée en

arrière-plan du pare-feu exécuté dans un data center d'entreprise, ou utilisée comme offre de service managée dans le cloud.

Avec le CaaS comme seule et unique fabric pour le data center et le cloud public, les entreprises peuvent développer des applications hybrides capables de connecter en toute sécurité leurs ressources internes au cloud public. Le CaaS contribue à impulser les déploiements sur le cloud hybride et le multicloud. Les développeurs et opérateurs peuvent facilement répartir les applications sur des environnements disparates.

Caractéristiques clés d'une plateforme de gestion de containers

Où qu'elle soit déployée, la plateforme de gestion des containers doit respecter les contraintes suivantes :

- **Homogénéité –** Les développeurs et opérateurs exigent une expérience homogène, dans le cloud public comme dans les environnements sur site.
- **Processus DevOps –** La plateforme doit intégrer les pratiques DevOps éprouvées afin de garantir un déploiement logiciel rapide.
- **Sécurité –** La plateforme garantira la sécurité de l'infrastructure et des applications. Elle doit détecter les vulnérabilités avant le déploiement applicatif tout en assurant un contrôle permanent de l'infrastructure afin d'identifier les éventuelles compromissions.
- **Fiabilité de l'infrastructure –** La plateforme doit fonctionner avec un modèle d'offre de services SLA garantissant une disponibilité maximale de l'infrastructure et de la plateforme.
- **Haute disponibilité des workloads –** Outre l'infrastructure, les applications métiers déployées sur la plateforme devront être hautement disponibles.
- **Observabilité –** La plateforme offrira une visibilité sur l'infrastructure, les ressources et les applications. Pour cela, elle doit s'appuyer sur la capture des indicateurs, évènements, journaux et traces de tout l'environnement, puis sur leur stockage dans un emplacement centralisé.

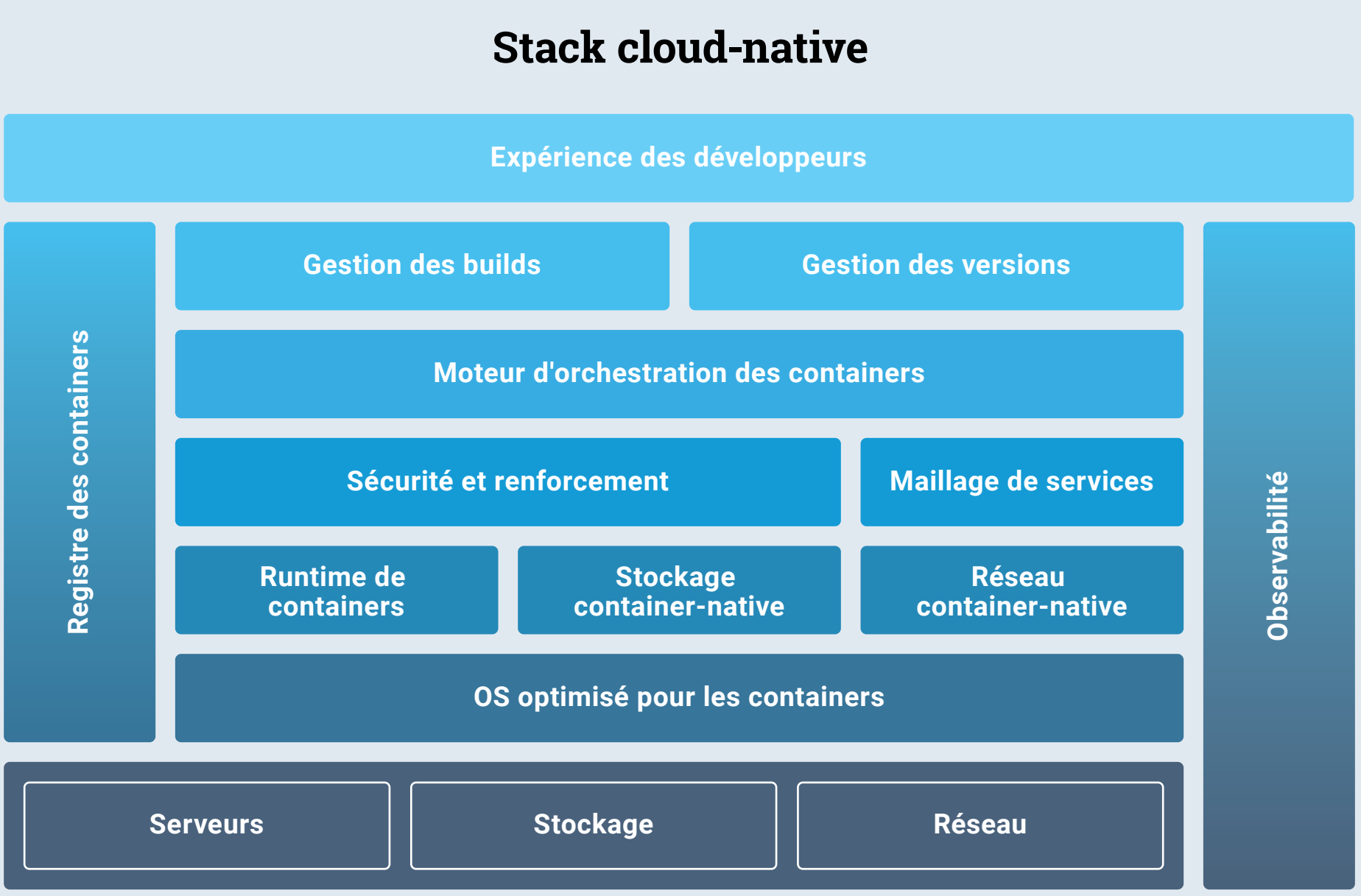
- **Solution multi-tenant et gestion régie par des politiques –** La plateforme de gestion des containers devra éventuellement assurer un isolement strict des locataires qui l'utilisent. La gouvernance du déploiement et de la gestion des applications se fera dans le respect de politiques bien définies.

Tour d'horizon des composants d'une plateforme de gestion de containers

Fourni sous la forme d'une plateforme intégrée de gestion des containers, un environnement cloud-native moderne se constitue de multiples éléments de base. Certains sont disponibles en tant que projets open source ou de solutions commerciales vendues par des éditeurs de logiciels indépendants.

La couche inférieure représente l'infrastructure physique du cluster sous la forme de composantes informatiques, de stockage et réseau. La plateforme rajoute plusieurs couches d'abstraction pour optimiser l'utilisation de l'infrastructure physique sous-jacente.

Figure 2.1 : *Stack cloud-native.*



Source : Janakiram MSV

Observons de plus près chacune de ces couches.

Système d'exploitation optimisé pour les containers

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Fedora CoreOS	Red Hat	Flatcar Container Linux	Non soumis
Talos	Talos Systems	RancherOS	Non soumis

Les containers redéfinissent le rôle des systèmes d'exploitation (OS). Avec le transfert de l'essentiel de la charge de travail vers les runtimes de containers, l'OS forme désormais une couche fine conçue pour offrir un accès aux ressources physiques. Cette évolution a donné naissance à des systèmes d'exploitation de nouvelle génération, appelés COS (Container-Optimized OS).

Comparés aux systèmes d'exploitation traditionnels, les COS se distinguent par leur légèreté et leur empreinte réduite. Ils intègrent les composantes nécessaires à l'exécution du runtime des containers. En choisissant le bon COS, vous contribuerez à optimiser le déploiement du CaaS.

Pour le déploiement de leur COS, les clients peuvent choisir entre Fedora CoreOS de Red Hat, Talos de Talos Systems, Flatcar Container Linux de Kinvolk GmbH. ou RancherOS de Rancher Labs (racheté par SUSE en juillet 2020).

La plupart des fournisseurs proposent un forfait d'abonnement optionnel incluant des mises à jour régulières, des correctifs et un support technique.

Runtime de containers

Le runtime de containers permet de gérer le cycle de vie d'un container. Il offre l'environnement d'exécution et sert d'interface entre le workload et le système d'exploitation hôte.

En 2015, The Linux Foundation a lancé l'Open Container Initiative (OCI) afin d'assurer la parité des implémentations de runtime de containers. L'OCI a déjà publié deux protocoles de spécification : l'un pour les environnements d'exécution (runtime-spec) et l'autre pour les formats d'images (image-spec).

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Docker Engine Enterprise	Mirantis	containerd	Graduated
--	--	CRI-O	Incubation
--	--	Docker-CE	Non soumis
--	--	Kata Containers	Non soumis
--	--	runC	Non soumis

Selon le site de l'OCI, la spécification d'exécution détaille l'exécution d'un « groupe de systèmes de fichiers » décompressé sur le disque. En substance, une implémentation OCI télécharge une image OCI, puis la décompresse dans un groupe de systèmes de fichiers de l'environnement d'exécution OCI.

La spécification du format d'image montre comment créer une image OCI – généralement par le biais d'un système de construction – et comment produire un manifeste d'image, un système de fichiers sérialisé (couche) et une configuration d'image.

Après avoir acquis Docker Enterprise, Mirantis rachète l'édition commerciale de Docker Engine (Docker Engine Enterprise), ce qui lui permet de proposer un support haut de gamme et des services professionnels.

Pour le runtime des containers, le projet containerd fait figure de standard. Ce projet de niveau « Graduated » de la CNCF est utilisé dans de nombreux environnements de production. Le projet d'incubation CRI-O de la CNCF bénéficie quant à lui de la participation active de la communauté Kubernetes.

Docker Engine (désormais appelé Docker-CE) est l'un des runtimes de containers les plus utilisés sur les plateformes de gestion de containers. Runtime de containers basé sur hyperviseur, Frakti (avec une approche plus ancienne) assure de son côté un isolement plus strict en exécutant des pods dans des machines virtuelles dédiées. Parmi les autres options existantes, on retrouve Kata Containers et runC.

Stockage container-native

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Red Hat OpenShift Container Storage	Red Hat	Ceph	Non soumis
Kubera	MayaData	Longhorn	Sandbox
Portworx	Portworx	OpenEBS	Sandbox
Robin	Robin Systems	Rook	Incubation
StorageOS	StorageOS	--	--
Trident	NetApp	--	--

Le stockage représente l'un des éléments les plus critiques des plateformes CaaS. Le stockage container-native expose les services de stockage sous-jacents aux containers et aux microservices. À l'instar du stockage défini par logiciel, il agrège et regroupe les ressources de stockage d'environnements disparates.

Le stockage container-native permet d'exécuter des workloads avec état dans les containers en fournissant des volumes persistants. Conjugué à des primitives Kubernetes tels que StatefulSets, ce type de stockage offre la fiabilité et la stabilité nécessaires à l'exécution des workloads critiques dans les environnements de production.

Si Kubernetes peut utiliser les systèmes de fichiers distribués traditionnels tels que le système de fichiers réseau (NFS) et GlusterFS, nous vous recommandons de faire appel à une fabric de stockage orientée container, conçue pour répondre aux exigences des workloads avec état exécutés en production. Les clients peuvent choisir parmi une gamme de projets open source et d'implémentations commerciales.

L'écosystème cloud-native a défini des spécifications de stockage via l'interface CSI (Container Storage Interface), favorisant ainsi une approche portable standard pour l'implémentation et la consommation des services de stockage par des workloads containerisés.

Ceph, Longhorn, OpenEBS et Rook constituent quelques exemples de projets open source de stockage container-native. De leur côté, Kubera par MayaData, Trident par NetApp, Portworx, Red

Hat OpenShift Container Storage, Robin par Robin System et StorageOS sont des offres commerciales assorties d'un support.

Les fournisseurs traditionnels tels que NetApp, Pure Storage et VMware fournissent également des plug-ins de stockage pour Kubernetes.

Relever les défis de l'infrastructure

Les offres managées de Kubernetes contribuent à réduire la complexité, mais aussi les compétences requises pour gérer les déploiements de containers à grande échelle. Au moment d'établir leurs feuilles de route technologiques, les professionnels IT accordent une importance particulière à la rationalisation de l'infrastructure exécutant les workloads Kubernetes.

Nous nous penchons ici sur les résultats de l'étude CNCF 2019 relative aux plans d'adoption Kubernetes actuels et futurs, ainsi que sur les défis de containerisation auxquels les utilisateurs Kubernetes sont confrontés. Les conclusions de ce rapport mettent en exergue l'impact des relations fournisseurs sur l'adoption précoce des technologies. Avec, toutefois, des retours partagés en termes de satisfaction.

Nombre d'utilisateurs Kubernetes ont intégré leurs fournisseurs de solutions de stockage et cloud actuels à leurs listes de candidats cloud-native. Il semble que ces utilisateurs aient éprouvé des difficultés à raccourcir ces listes. Au moins 5 % des utilisateurs Kubernetes ont évalué les 38 choix proposés par l'étude.

Avec l'émergence du Kubernetes managé, les fournisseurs cloud proposent un stockage par blocs au travers de classes de stockage et d'un provisionnement dynamique. Les clients peuvent ainsi raccorder des volumes Amazon Elastic Block Store (EBS) à AWS, aux disques managés Azure, à Google Persistent Disks et aux nœuds worker Kubernetes exécutés dans AWS, GCP et Microsoft Azure. Une situation avantageuse pour les fournisseurs cloud.

Amazon EBS, Google Persistent Disk et Azure Disk Storage se classent en tête des solutions de stockage cloud-native préférées des utilisateurs Kubernetes, comme le révèle l'étude (cf. figure 2.2 ci-dessous). Dans de nombreux cas, StatefulSets a permis aux workloads de cluster d'accéder au

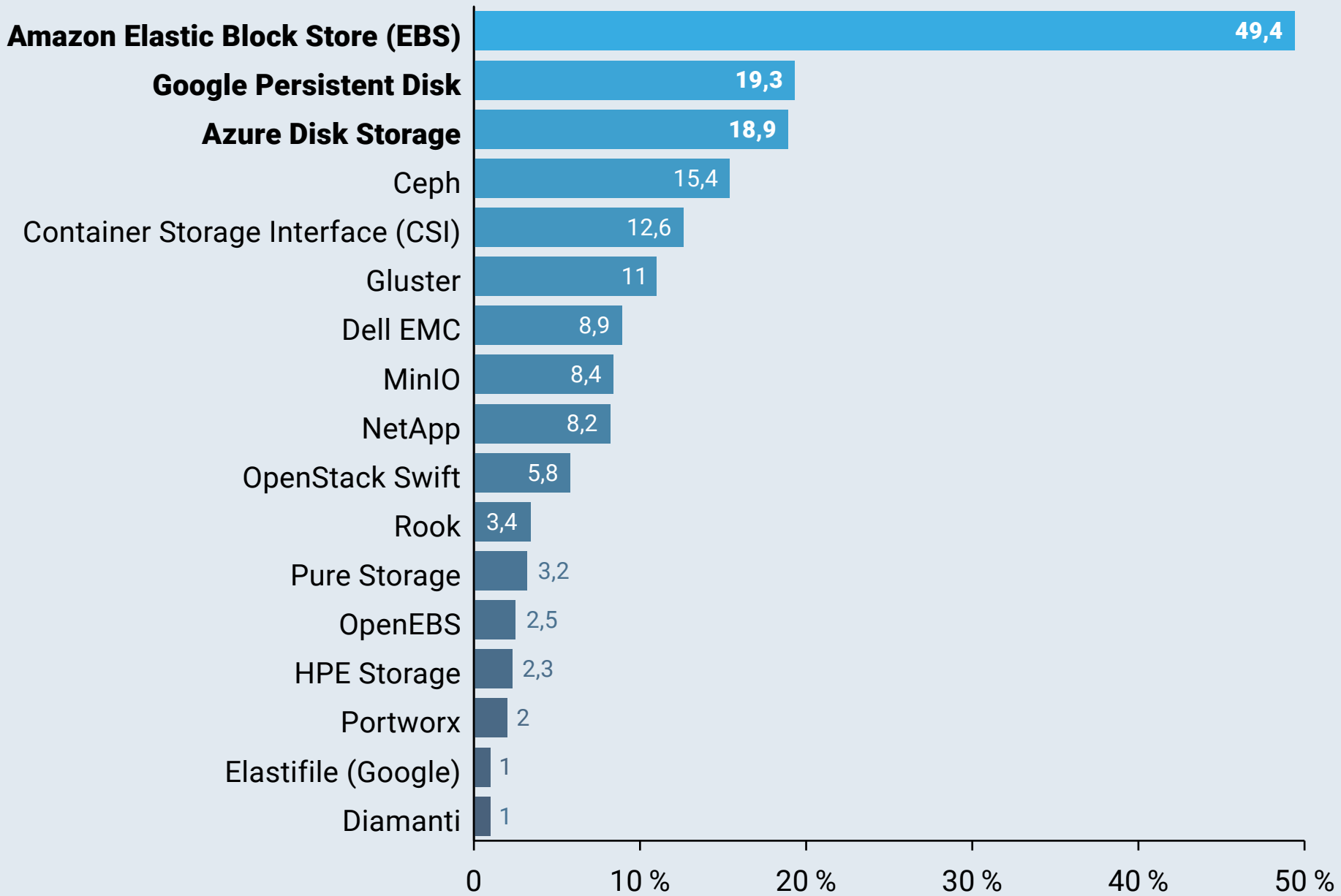


Figure 2.2 : *Les systèmes de fichiers tels que Ceph sont souvent perçus comme faisant concurrence au stockage cloud, par rapport aux offres des entreprises de stockage traditionnelles.*

stockage par blocs proposé par le fournisseur cloud. Bien qu'il soit largement adopté, le stockage par blocs proposé par les grands fournisseurs cloud n'a pas été conçu spécifiquement pour les workloads Kubernetes.

Viennent ensuite Ceph, CSI et Gluster, avec 37 % des utilisateurs Gluster exploitant également Ceph. Ceph et Gluster sont des systèmes de fichiers distribués qui ajoutent une couche de persistance sur plusieurs nœuds. Assez mal intégrés aux outils et workflows Kubernetes, ces systèmes peuvent néanmoins poser aux administrateurs de stockage des problèmes de maintenance et de configuration.

Plus bas dans le classement, on retrouve certaines entreprises orientées stockage bien établies telles que Dell EMC, NetApp et Pure Storage. À l'origine, Kubernetes avait intégré des plug-ins de

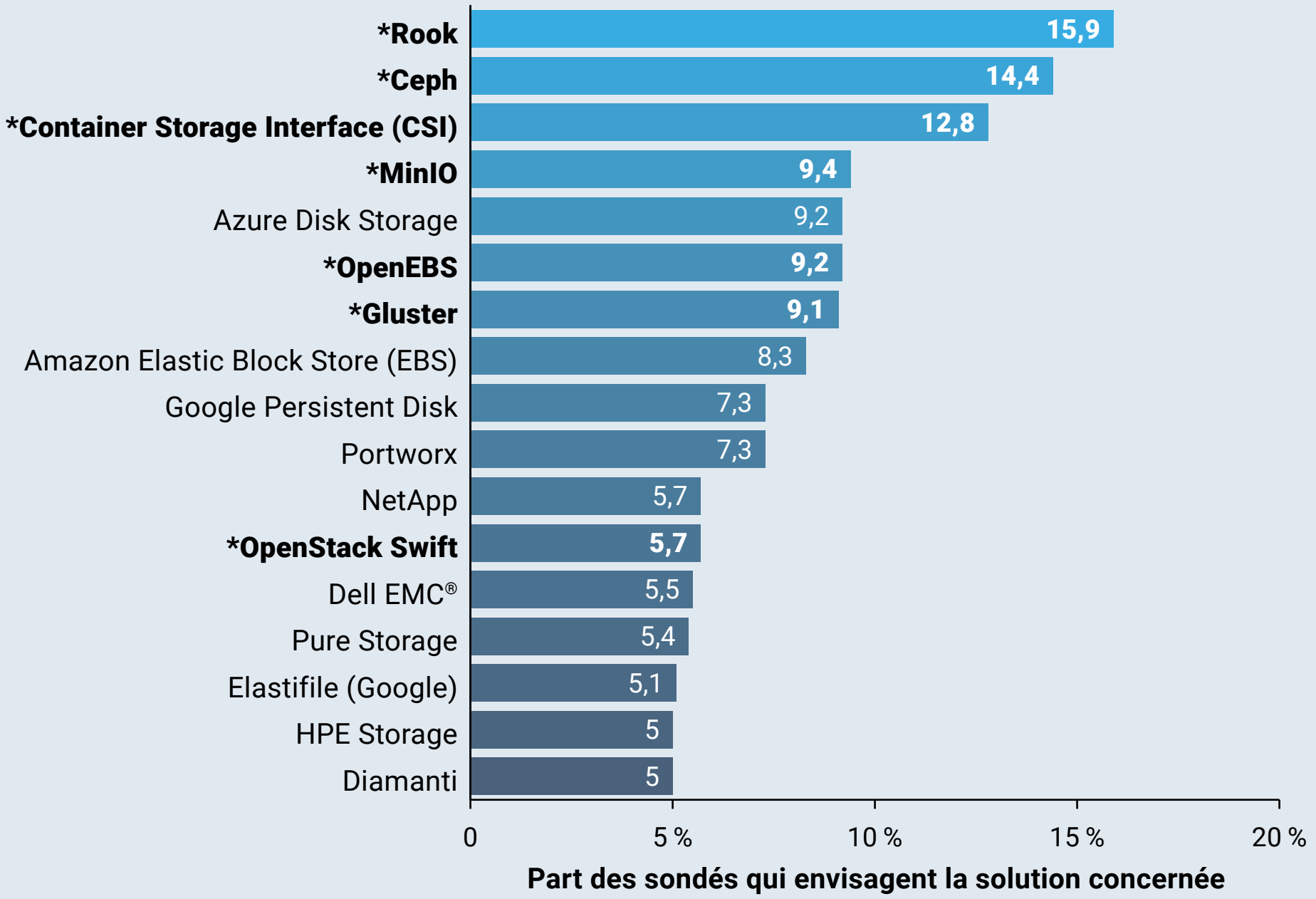
volume pour se connecter aux back-ends de stockage de ces entreprises. Malheureusement, avec l'encombrement de la distribution Kubernetes de base, la moindre mise à jour ou modification de plug-in nécessitait de devoir recréer et compiler tout le code.

Les clients des entreprises de stockage traditionnelles ont été plus enclins à rapporter des problèmes de stockage. Ainsi, 46 % des clients Pure Storage ont éprouvé des difficultés à gérer le stockage lié aux containers, contre seulement 27 % des utilisateurs Kubernetes. Lueur d'espoir à l'horizon, 13 % des sondés ont utilisé l'interface CSI (Container Storage Interface). Adopté par Kubernetes en 2019, CSI élimine la nécessité d'une intégration upstream permanente. Les éditeurs de solutions de stockage traditionnels, les fournisseurs cloud et les entreprises spécialisées dans le stockage de containers telles que Portworx transitent vers CSI.

L'option CSI a été étudiée de près par les entreprises qui cherchaient à résoudre leurs problématiques de stockage de containers. Si 13 % de l'ensemble des utilisateurs Kubernetes ont envisagé d'adopter l'interface CSI, ce chiffre grimpe à 22 % pour les utilisateurs confrontés à des problèmes de stockage.

Bien que certains utilisateurs en quête de nouvelles options de stockage aient envisagé de se tourner vers des entreprises établies, une majorité d'entre eux étaient davantage intéressés par des projets open source (cf. figure 2.3). Par rapport à la moyenne générale, les 26,6 % des utilisateurs Kubernetes confrontés à des problèmes de stockage étaient plus susceptibles d'opter pour Rook (25,9 % contre 15,9 %), Ceph (23,4 % contre 14,4 %), Gluster (14,6 % contre 9,1 %), OpenEBS (14,6 % contre 9,2 %) et MinIO (12,8 % contre 9,4 %). Cette approche open source n'était pas motivée par un besoin de vendre du matériel.

Pour les entreprises de stockage traditionnelles comme pour les solutions de stockage cloud-native nouvelle génération, les utilisateurs avaient plus tendance à évoquer des problèmes de stockage. La mise en place de nouvelles approches (CSI, etc.) a toutefois permis aux entreprises de stockage traditionnelles de résoudre les problèmes de leurs clients. Si les utilisateurs des nouvelles offres — telles que MayaData OpenEBS, Minio et Portworx — ont souvent indiqué avoir été confrontés à des défis de stockage, leur problématique concernait souvent la connexion des datastores existants.



Source : analyse de l'enquête 2019 de la CNCF par The New Stack. Q. Veuillez préciser si votre entreprise/organisation envisage d'utiliser ou utilise déjà l'un de ces projets de stockage cloud-native en production. Quels sont les problèmes que soulèvent l'utilisation ou le déploiement de containers ? Plusieurs réponses possibles. Seules les offres utilisées par au moins 1 % des sondés avec au moins un cluster Kubernetes figurent sur ce graphique. n=1149.

Figure 2.3 : *Une analyse approfondie révèle que les utilisateurs Kubernetes ayant signalé des problèmes de stockage (données non affichées) ont une impression 50 % plus positive de Rook, Ceph et OpenEBS – trois opérateurs utilisant une interface CSI.*

Les primo-adoptants de solutions avancées tierces ont fréquemment été confrontés à des problèmes d'implémentation. À terme, il sera intéressant d'évaluer l'efficacité des nouveaux acteurs, ainsi que leur capacité à capter la clientèle des entreprises cloud et de stockage traditionnelles.

Réseau container-native

Tout comme le stockage container-native, le réseau container-native gère l'abstraction de l'infrastructure réseau physique afin de proposer un réseau « à plat » de containers. Ce type de réseau s'intègre parfaitement à Kubernetes afin de résoudre les problématiques pod-to-pod, node-to-node, pod-to-service et de communication externe.

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Calico Enterprise	Tigera	Cilium	Non soumis
Weave Net (contacter l'éditeur pour l'abonnement Enterprise)	Weaveworks	Contiv	Non soumis
--	--	Flannel	Non soumis
--	--	Project Calico	Non soumis
--	--	Tungsten Fabric	Non soumis
--	--	Weave Net	Non soumis

Kubernetes prend en charge une variété de plug-ins basés sur la spécification CNI (Container Network Interface), laquelle définit la connectivité réseau des containers et gère les ressources réseau une fois le container effacé. Le projet CNI fait partie des projets d'incubation de la CNFC.

Au-delà d'une simple connectivité de base, les réseaux container-native assurent une application dynamique des règles de sécurité réseau. L'utilisation d'une politique prédéfinie permet ainsi de configurer un contrôle granulaire des communications entre containers, pods et nœuds.

Il est essentiel de choisir le bon environnement réseau pour maintenir et protéger la plateforme CaaS. Les clients peuvent se tourner vers des projets open source comme Cilium, Contiv, Flannel, Project Calico, Tungsten Fabric et Weave Net. Du côté des solutions commerciales, Tigera offre Calico Enterprise tandis que Weaveworks propose un abonnement à Weave Net.

Les offres managées CaaS proposées par les fournisseurs de cloud public s'intègrent parfaitement à l'environnement réseau virtuel existant. AWS dispose par exemple d'un plug-in CNI pour Amazon Elastic Kubernetes Service (EKS) basé sur Amazon Virtual Private Cloud (VPC). De son côté, Microsoft a créé le plug-in d'interface réseau de container (CNI) de réseau virtuel Azure Virtual Network Container Networking Interface pour Azure Kubernetes Service (AKS).

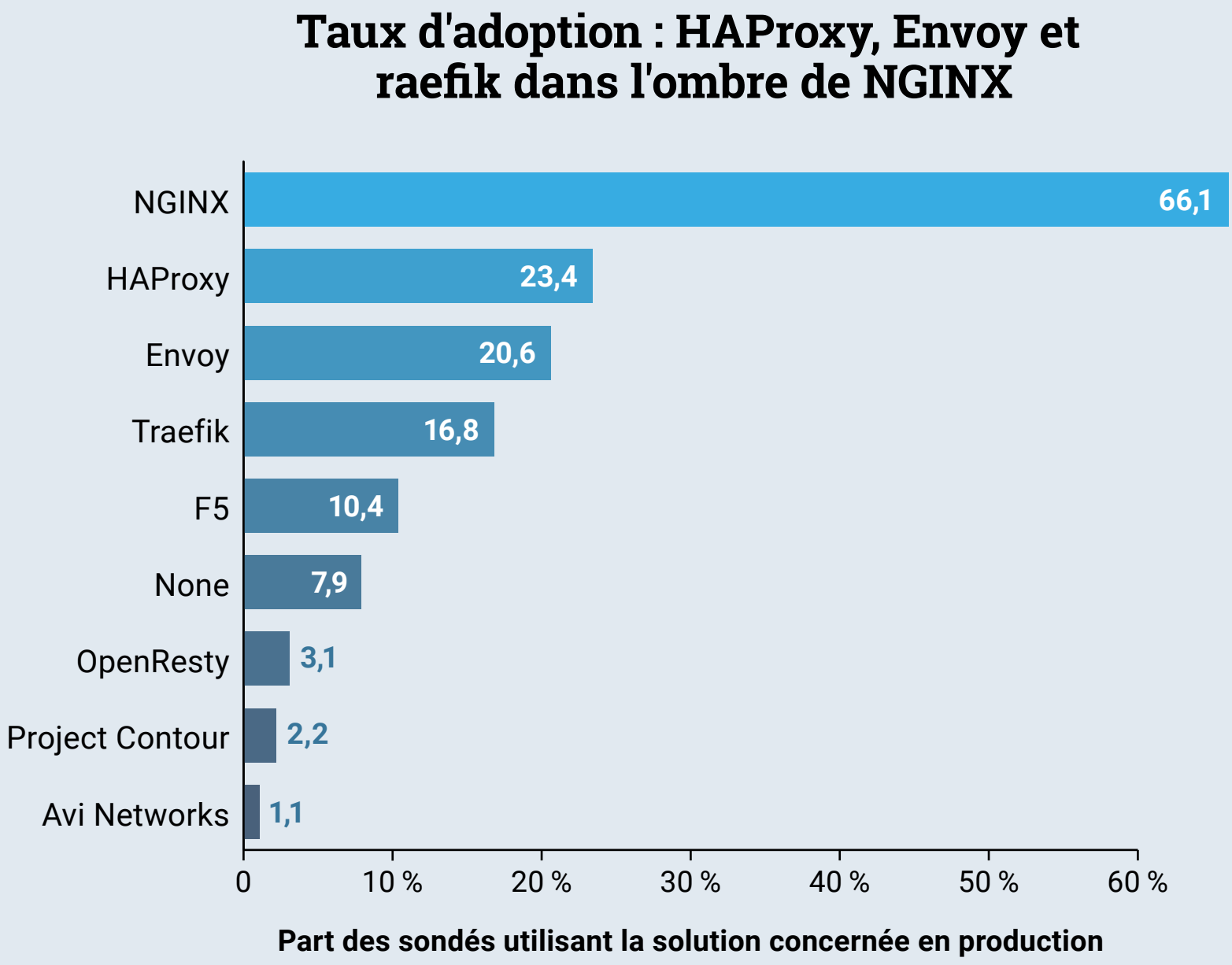
Intéressons-nous à présent aux données réseau de l'étude CNCF 2019.

La problématique réseau a perdu de son importance au fil des années, bien que les utilisateurs Kubernetes continuent d'évaluer les fournisseurs de solution Ingress. Ainsi, si chaque utilisateur

Kubernetes recourt à 1,5 fournisseur Ingress en moyenne, 28 % des sondés ayant été confrontés à des problématiques réseau faisaient appel à trois fournisseurs en moyenne. Présent dans 66 % des environnements Kubernetes, NGINX ne permettait pas en soi de répondre aux attentes de tous les utilisateurs. Moins utilisés, HAProxy et Envoy ont néanmoins vu leur taux d'adoption grimper parmi les utilisateurs confrontés à des problématiques réseau.

Pour résoudre leurs problèmes, les utilisateurs semblent ainsi se tourner vers les nouveaux entrants. À l'avenir, les solutions devraient se démarquer les unes des autres par les protocoles gérés et l'intégration ou non d'une passerelle API.

Figure 2.4 : *Même avec une solution Ingress, l'utilisation de NGINX avec Kubernetes a un faible impact sur la résolution des problématiques réseau.*



Source : analyse de l'enquête 2019 de la CNCF par The New Stack. Q. Quel(s) fournisseur(s) de solutions Ingress (ou proxy de services) utilisez-vous ? Plusieurs réponses possibles. Quels sont les problèmes que soulèvent l'utilisation ou le déploiement de containers ? Plusieurs réponses possibles. Seules les offres utilisées par au moins 1 % des sondés avec au moins un cluster Kubernetes figurent sur ce graphique. n=1197.

Sécurité et renforcement

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Aqua	Aqua Security Software	Clair	Non soumis
Calico Enterprise	Tigera	Falco	Incubation
Prisma Cloud	Palo Alto Networks	Open Policy Agent (OPA)	Incubation
Snyk	Snyk	Snyk (Open Source)	Non soumis
StackRox	StackRox	--	--

La protection des plateformes de gestion de containers passe par un renforcement du réseau, une analyse des images de containers, ainsi qu'une détection des menaces et un profilage des risques réguliers.

La sécurité du réseau s'inscrit au cœur du durcissement de la plateforme. Les équipes informatiques doivent investir dans une couche réseau « Zero Trust » afin de satisfaire aux exigences de sécurité et de conformité. Ces réseaux appliquent un contrôle granulaire des politiques sur plusieurs points de l'infrastructure, recherchent les anomalies et chiffrent et autorisent le trafic entre les microservices utilisant des protocoles sécurisés tels que mTLS (mutual Transport Layer Security). Calico Enterprise de Tigera est l'un des principaux fournisseurs réseau « Zero Trust » pour les plateformes Kubernetes et CaaS.

Les plateformes de containers sécurisées s'appuient sur les techniques suivantes pour protéger l'infrastructure et les applications :

- Utilisation d'**images de confiance** signées et vérifiées. Cette fonctionnalité est intégrée au composant du registre de containers qui stocke les images du container.
- Mise en place d'un **stockage chiffré** permettant de stocker et de retrouver en toute sécurité les données confidentielles (identifiants, mots de passe, etc.).
- Intégration de la plateforme au protocole LDAP (Lightweight Directory Access Protocol) et au système Active Directory (AD) pour une configuration du **contrôle intégré des accès basé sur les rôles** (RBAC).

Aqua, Prisma Cloud et StackRox proposent des solutions commerciales pour la protection des containers et de l'infrastructure Kubernetes. Snyk offre de son côté des solutions commerciales et open source.

Les projets open source comme Clair effectuent des analyses statiques des vulnérabilités dans les containers d'application.

Falco, un projet d'incubation de la CNCF, est un moniteur d'activité comportementale conçu pour détecter les activités anormales dans les applications cloud-native. Falco fournit un audit du système au tout premier niveau : le noyau. Puis il enrichit les données avec d'autres flux d'entrée (indicateurs de runtimes de containers, indicateurs Kubernetes, etc.) afin d'alerter les utilisateurs dès qu'une règle prédéfinie est atteinte.

Open Policy Agent (OPA), un autre projet d'incubation de la CNCF, offre un ensemble d'outils et un framework unifié pour les politiques sur tout l'environnement. OPA propose un langage déclaratif de haut niveau qui permet aux développeurs et aux opérateurs de spécifier une politique sous forme de code, ainsi que des API simples pour décharger le logiciel des processus décisionnels liés aux politiques. OPA peut appliquer des politiques pour les microservices, Kubernetes, les pipelines CI/CD, les passerelles API, et plus encore.

Maillage de services

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Aspen Mesh	F5	Consul	Non soumis
Consul Enterprise	HashiCorp	Contour	Incubation
Grey Matter	Decipher Technology Studios	Envoy	Graduated
Kong Enterprise	Kong	Istio	Non soumis
Linkerd Commercial Support	Buoyant	Kuma	Sandbox
Maesh	Containous	Linkerd	Incubation
Service Mesh Hub	Solo.io	Service Mesh Interface (SMI)	Sandbox
Tetrate	Tetrate	Zuul	Non soumis

Le maillage de services devient un élément central de l'environnement cloud-native. Cette fonctionnalité offre un contrôle granulaire du trafic des différents microservices tout en permettant d'analyser l'intégrité de chaque microservice.

Elle vient compléter le réseau container-native en se focalisant sur le trafic est-ouest, tandis que la couche de cœur de réseau gère le trafic nord-sud. Le maillage de services ajoute un proxy à chaque microservice qui intercepte le trafic entrant et sortant. Placé à proximité du microservice, celui-ci peut suivre l'état du service.

Consul, Contour, Envoy, Istio, Kuma, Linkerd, Service Mesh Interface (SMI) et Zuul figurent parmi les projets de maillage de services open source les plus plébiscités.

Les clients peuvent faire leur choix parmi des implémentations commerciales telles que Aspen Mesh de F5, Consul Enterprise de HashiCorp, Grey Matter de Decipher Technology Studios, Kong Enterprise de Kong, Linkerd Commercial Support de Buoyant, Maesh de Containous, Service Mesh Hub de Solo.io et Tetrate.

Penchons-nous à présent sur les données de l'étude CNCF 2019 portant sur le maillage de services.

Les résultats obtenus montrent que 27 % des utilisateurs Kubernetes ont été confrontés à des problèmes de maillage de services – tant en termes de choix d'outils que d'opérations quotidiennes.

HashiCorp Consul et Istio atteignent pratiquement le même niveau d'utilisation auprès des utilisateurs Kubernetes. Toutefois, les utilisateurs de Consul déployaient probablement déjà cette solution pour la découverte de service, et ne faisaient de ce fait que tester le cas d'usage lié au maillage de services.

Pour Istio et Linkerd, la situation était tout autre. Ces deux solutions ont fait l'objet d'un nombre d'évaluations nettement supérieur de la part des utilisateurs ayant eu des difficultés avec cette fonctionnalité.

La très honorable quatrième place de Netflix Zuul montre le degré d'intérêt des utilisateurs pour

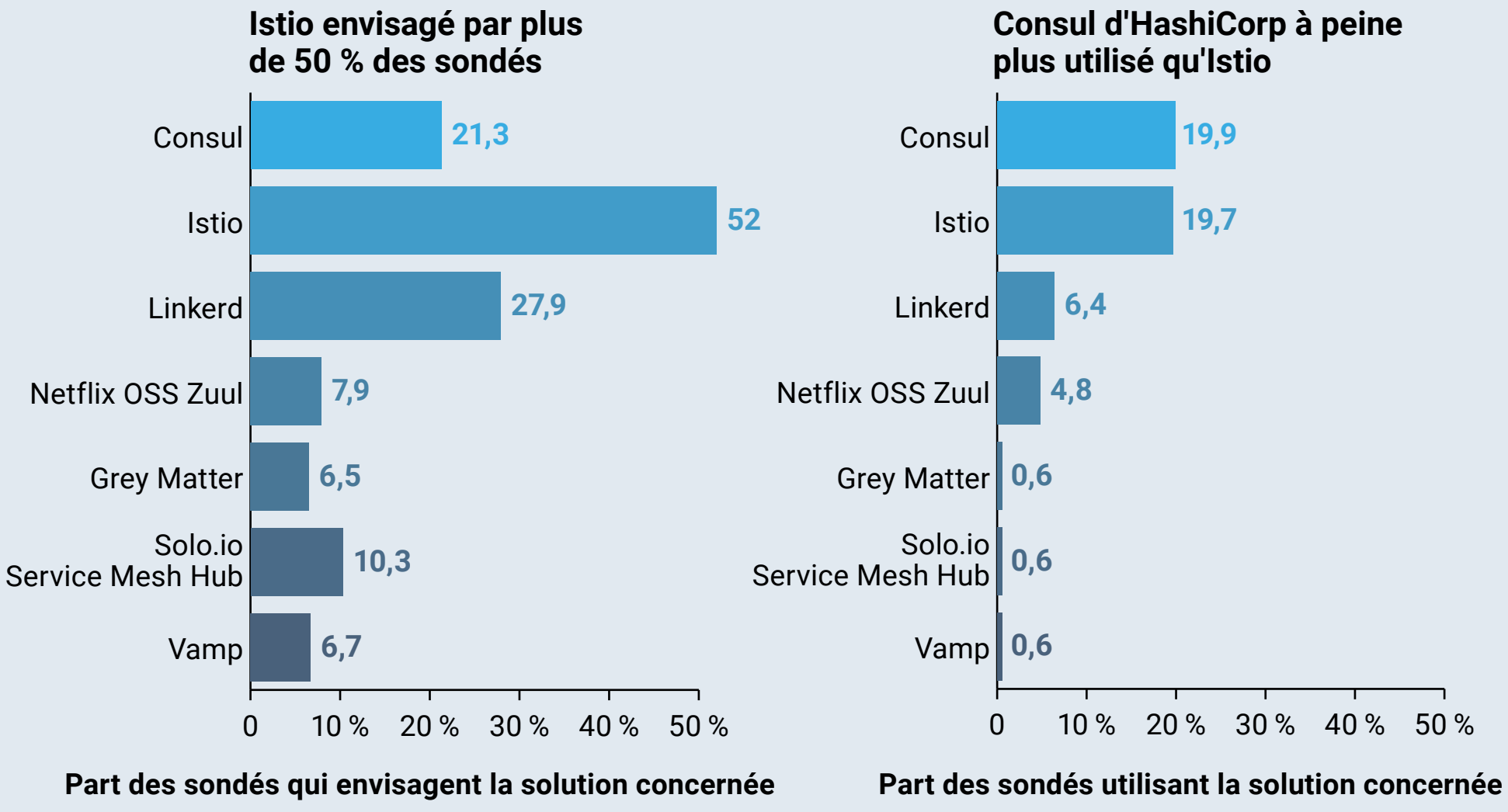


Figure 2.5 : *Consul est utilisé par une base de développeurs pour la découverte de services, et donc très certainement aussi pour le maillage de services associé. Parallèlement, plus de la moitié des utilisateurs Kubernetes évaluent Istio pour la première fois.*

cette solution. Fait intéressant, seuls 23 % des utilisateurs Zuul estiment avoir été confrontés à des problèmes de maillage de services, contre 27 % pour l'ensemble des utilisateurs. Du fait d'une notoriété inférieure à celle d'Istio, Zuul a probablement été choisi pour ses capacités reconnues à répondre à un besoin spécifique.

Grey Matter, Service Mesh Hub et Vamp n'ont enregistré qu'une utilisation résiduelle. Toutefois, leur notoriété leur a permis d'être présents sur les listes.

Si l'on regarde vers l'avenir, le principal défi auquel les utilisateurs du maillage de services seront confrontés devrait être celui de son intégration avec leur environnement Kubernetes. D'ici là, Istio, Linkerd et d'autres solutions feront tout pour améliorer leur intégration avec Envoy.

Moteur d'orchestration des containers

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Canonical Distribution of Kubernetes (CDK)	Canonical	Kubernetes	Graduated
Cisco Container Platform	Cisco	OKD	Non soumis
Docker Enterprise	Mirantis	Rancher Kubernetes Engine (RKE)	Non soumis
HPE Ezmeral Container Platform	HPE	--	--
Mesosphere Kubernetes Engine (MKE)	D2IQ	--	--
Mirantis Cloud Platform	Mirantis	--	--
Rancher Kubernetes Engine (RKE)	Rancher Labs	--	--
Red Hat OpenShift	Red Hat	--	--
SUSE CaaS Platform	SUSE	--	--
Tanzu Kubernetes Grid	VMware	--	--

Comme nous l'évoquions déjà dans le chapitre 1, Kubernetes est le moteur d'orchestration de containers le plus couramment utilisé sur les plateformes CaaS.

Bien que Kubernetes puisse être déployé depuis la distribution open source de base disponible sur GitHub, les clients pourront vouloir investir dans une distribution commerciale livrée avec des packs de maintenance et de support. Canonical Distribution of Kubernetes, Cisco Container Platform, Docker Enterprise, HPE Ezmeral Container Platform, Mesosphere Kubernetes Engine, Mirantis Cloud Platform, Rancher Kubernetes Engine, Red Hat OpenShift (également utilisable en tant que PaaS), SUSE CaaS Platform et VMware Tanzu Kubernetes Grid font partie des distributions commerciales actuellement proposées sur le marché.

Les projets open source OKD et Rancher Kubernetes Engine (RKE) sont également disponibles.

Les clients ayant investi dans des plateformes CaaS de cloud public basées sur Google Kubernetes

Engine ou IBM Kubernetes Service peuvent choisir entre Anthos et IBM Cloud Paks, deux solutions parfaitement intégrées au plan de contrôle exécuté dans le cloud.

Dans le chapitre précédent, nous avons passé en revue les modèles de déploiement Kubernetes.

Registre des containers

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Docker Hub	Docker	Docker Registry	Non soumis
Docker Trusted Registry	Mirantis	Harbor	Graduated
GitLab Container Registry	GitLab	Project Quay	Non soumis
JFrog Container Registry / Artifactory	JFrog	--	--
Red Hat Quay	Red Hat	--	--

Les microservices sont packagés sous forme d'images de containers avant d'être déployés et mis à l'échelle par le moteur d'orchestration. Le registre de containers fait office de référentiel central sur lequel sont stockées les images de containers. Son intégration aux outils de gestion de builds lui permet de générer automatiquement des images à chaque nouvelle version du service. Un outil de gestion des lancements sélectionne la version la plus récente de l'image dans le registre avant de la déployer.

Docker Trusted Registry de Mirantis, GitLab Container Registry, JFrog Container Registry et Red Hat Quay sont quelques-uns des registres de containers disponibles, en plus de ceux proposés par les fournisseurs de solutions cloud. Docker offre également un registre hébergé appelé Docker Hub, de même qu'un registre open source dénommé Docker Registry.

En plus de stocker les images de containers, ces produits effectuent une analyse de vulnérabilités sur les images et gèrent l'authentification et l'autorisation des utilisateurs. Il arrive que des entreprises souhaitent conserver leurs registres sur site et, de ce fait, désactivent l'accès aux registres en ligne. Pour répondre à leurs attentes, certaines offres commerciales intègrent des

registres internes. Une autre solution consiste à créer un registre interne avec plusieurs de ces mêmes fonctionnalités.

Les fournisseurs cloud proposant un service Kubernetes managé offrent la possibilité d'héberger un registre de containers privé au sein du compte client. La colocalisation du cluster et du registre dans une même région et sur un même compte contribue à réduire la latence et à renforcer la sécurité.

Le registre d'images de containers open source Harbor, initialement développé par VMware, protège les images grâce à un contrôle d'accès basé sur les rôles, analyse les images afin de détecter les éventuelles vulnérabilités et confirme l'authenticité des images. Project Quay, la distribution open source signée Red Hat Quay, propose une interface utilisateur web grand public, une recherche des vulnérabilités images, et un stockage et une protection des données d'entreprise.

Gestion des builds

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
CircleCI	Circle Internet Services	GitLab	Non soumis
CloudBees CI	CloudBees	Jenkins / Jenkins X	Non soumis
Digital.ai	Digital.ai Software	--	--
GitHub Actions	GitHub	--	--
GitLab CI	GitLab	--	--
JFrog Pipelines	JFrog	--	--
SemaphoreCI	Rendered Text	--	--
Travis CI	Travis CI	--	--

Pour assurer un déploiement rapide des microservices, chaque code de temps est intégré au référentiel. Une nouvelle image de container est créée, puis rajoutée au registre de containers. Ces images peuvent alors être déployées dans les environnements de préproduction ou de test au sein du CaaS pour la réalisation de tests automatisés ou manuels.

La gestion des builds implique de convertir la version la plus récente du code en différents artefacts, et notamment en images de containers, bibliothèques et fichiers exécutables. Il s'agit d'une étape importante du pipeline CI/CD (intégration continue / déploiement continu).

Le système de gestion du code source, basé sur des référentiels Git internes ou externes, active un processus de création automatisé et sécurisé qui donnera lieu à un artefact de déploiement. Le résultat de cette étape peut inclure des images de containers, des graphiques Helm et des déploiements Kubernetes.

GitLab et Jenkins sont des logiciels prisés de gestion de builds que l'on retrouve en version open source ou commerciale. CloudBees CI est une version commerciale du serveur de gestion de builds Jenkins.

CircleCI, Digital.ai, GitHub Actions, JFrog Pipelines, SemaphoreCI et Travis CI sont de leur côté des solutions CI/CD commerciales conçues pour les microservices.

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
Armory Enterprise Spinnaker	Armory	**Argo CD**	Incubation
Open Enterprise Spinnaker	OpsMx	**Flux**	Sandbox

Gestion des versions

Ultime étape du pipeline CI/CD, la gestion des versions nécessite de déployer une version entièrement testée des microservices dans l'environnement de production. Les stratégies de déploiement avancées telles que les versions de type « canary », les déploiements Blue-Green, les rollbacks et les roll forwards constituent une partie de la gestion des versions.

Spinnaker est l'un des principaux outils de gestion des lancements de microservices. Il complète Jenkins et les autres outils de gestion de builds en automatisant la gestion et le déploiement des artefacts. Armory et OpsMx proposent des implémentations commerciales de Spinnaker.

GitOps est une technologie émergente conçue pour implémenter Configuration as Code (CaC).

L'artefact YAML et les graphiques Helm sont des versions contrôlées et maintenues dans le référentiel Git. À l'aide d'une stratégie pull ou push, les modifications de configuration sont appliquées au cluster. Argo CD (projet d'incubation de la CNCF) et Flux (projet de sandbox de la CNCF) sont deux projets open source très utilisés pour la configuration de GitOps.

Produits commerciaux	Éditeur	Projets open source	Statut CNCF
AppDynamics	Cisco	Fluentd	Graduated
Datadog	Datadog	Grafana	Non soumis
Dynatrace	Dynatrace	Jaeger	Graduated
Honeycomb	Honeycomb	OpenTelemetry	Sandbox
Lightstep	Lightstep	OpenTracing	Incubation
New Relic One	New Relic	Prometheus	Graduated
Sysdig Monitor	Sysdig	--	--

Observabilité

L'observabilité, c'est l'art de saisir les indicateurs, journaux, évènements et traces de tout l'environnement.

Le monitoring permet de collecter les indicateurs de l'infrastructure et de l'environnement applicatif de la plateforme. Avec une plateforme de monitoring robuste, il est possible de surveiller l'état du cluster Kubernetes, mais aussi les applications déployées. Les agents installés dans chaque nœud recueillent des informations détaillées sur l'utilisation des ressources, l'intégrité du cluster et du nœud, la disponibilité de stockage et de mémoire, le nombre de containers exécutés sur chaque nœud et des indicateurs détaillés sur les pods. Les clients peuvent déployer des outils de monitoring open source tels que Grafana et Prometheus, ou se tourner vers des plateformes commerciales comme DataDog, Dynatrace, New Relic One et Sysdig Monitor.

La journalisation collecte et agrège les informations, les alertes et les erreurs soulevées par les différents composants de la plateforme cloud-native. Presque tous les composants de Kubernetes génèrent des journaux offrant une analyse détaillée de l'état actuel du cluster. Dans le cadre de

leurs bonnes pratiques, les développeurs sont incités à intégrer la journalisation au niveau des microservices. Plusieurs agents sont déployés dans le cluster pour collecter et diffuser les journaux vers un référentiel central. Outil de collecte de données open source particulièrement prisé, Fluentd peut être utilisé en combinaison avec Elastic et Kibana pour la visualisation et la recherche de journaux. Les logiciels de maillage de services tels qu'Istio et Linkerd sont étroitement intégrés aux plateformes de journalisation.

Prometheus et Fluentd sont l'un comme l'autre des projets « Graduated » de la CNCF.

Dans la mesure où les applications cloud-native sont assemblées à partir de services disparates et autonomes, il est important de suivre la chaîne de communication et le délai de réponse de chaque service. Cette approche permet de surveiller et d'analyser les performances des applications.

Les outils open source tels que Jaeger et OpenTracing peuvent vous fournir des fonctionnalités APM (Application Performance Management) pour les microservices. Parmi les offres commerciales, vous pouvez faire votre choix entre AppDynamics, Honeycomb et Lightstep – ces deux dernières solutions proposant des fonctionnalités de suivi et de monitoring de bout en bout pour les nouvelles applications.

OpenTelemetry offre un ensemble d'API, de bibliothèques, d'agents et de collecteurs de services pour capturer les traces et les indicateurs distribués des applications cloud-native.

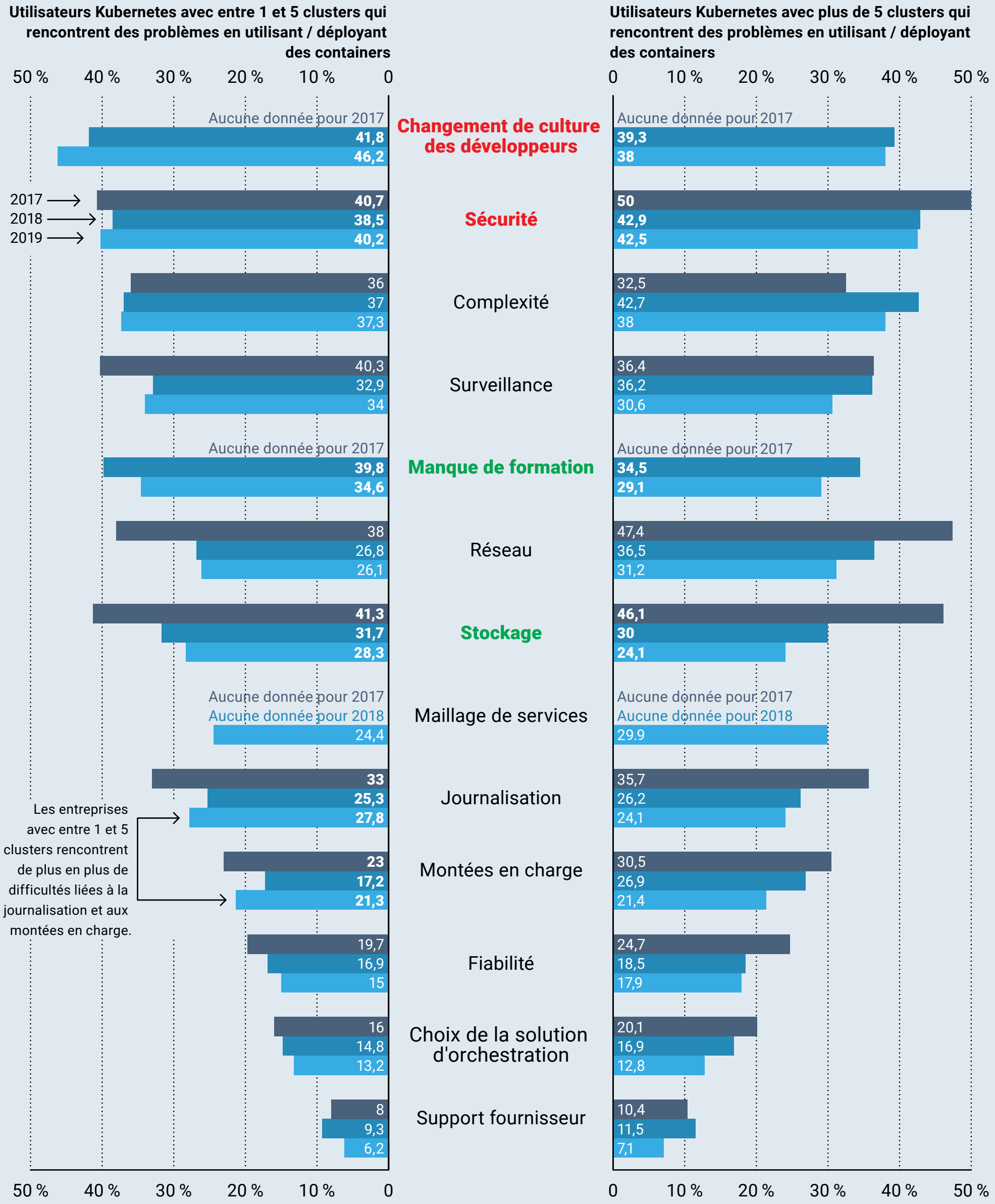
OpenTracing est un projet d'incubation de la CNCF, tandis qu'OpenTelemetry s'inscrit dans la catégorie « Sandbox ».

Containers : problématiques des utilisateurs Kubernetes

Revenons aux données des études CNCF réalisées entre 2017 et 2019 et penchons-nous sur les défis liés aux containers des utilisateurs Kubernetes.

Les entreprises disposant de plus de cinq clusters ont vu une nette diminution de leurs problématiques containers en 2018. Et la tendance s'est poursuivie en 2019. Pour les utilisateurs

Réduction des problèmes de formation et de stockage, mais persistance des difficultés liées à la sécurité et au changement de culture



Source : analyse des enquêtes 2017, 2018 et 2019 de la CNCF par The New Stack. Q. Quels sont les problèmes que soulèvent l'utilisation ou le déploiement de containers ?Plusieurs réponses possibles. Q. Quels sont les problèmes que soulèvent l'utilisation ou le déploiement de containers ? Plusieurs réponses possibles.Entre 1 et 5 clusters Kubernetes : 2017, n=300; 2018, n=925; 2019, n=729. Plus de 5 clusters Kubernetes : 2017, n=154; 2018, n=550; 2019, n=468.

Figure 2.6 : *Les utilisateurs Kubernetes disposant de 1 à 5 clusters sont confrontés à des problématiques culturelles avec l'équipe de développement. Ceux ayant plus de 5 clusters ont observé une légère baisse de toutes les problématiques en 2019.*

Kubernetes possédant un à cinq clusters, la sécurité, la complexité et le changement culturel induit pour l'équipe de développement représentent les principaux défis en termes d'utilisation ou de déploiement des containers.

Selon la CNCF, les entreprises tendent à résoudre les problématiques de réseau et de stockage des données indépendamment du nombre de clusters utilisés. La maturité des spécifications CSI et la présence d'une interface CSI dans la plupart des solutions de stockage pourraient expliquer la résolution progressive des problématiques de stockage. Si le réseau n'a généralement pas constitué une source de préoccupation majeure, c'est peut-être parce que les technologies CNI (Container Networking Interface) comme Flannel se sont imposées comme le standard reconnu pour la communication des containers dans les clusters.

La baisse continue des plaintes des utilisateurs portant sur le manque de formation ou de support fournisseur montre le succès de la CNCF, de KubeCon + CloudNativeCon et de l'écosystème open source en général.

Les problèmes d'évolutivité des déploiements ont été de plus en plus rares dans les entreprises comptant plus de cinq clusters. L'écosystème Kubernetes a permis d'effectuer une montée en charge des workloads, mais aussi de les surveiller et de les journaliser dans des clusters plus larges.

Les entreprises avec un maximum de cinq clusters ont réalisé de moins bonnes performances avec une hausse des difficultés liées au changement d'échelle des déploiements, à la culture du développement et à la journalisation en 2019.

Dans les petites entreprises qui n'ont pas été aussi loin dans le réalignement de leur structure organisationnelle, le changement culturel de l'équipe de développement a représenté un défi de taille. Bien que nombre de sociétés ont formé des équipes DevOps et SRE, leurs développeurs ont continué à devoir avant tout gérer les différents aspects liés à la couche d'infrastructure.

Plateforme de gestion de containers d'entreprise

Les fournisseurs de plateformes et de systèmes d'exploitation ont créé des solutions commerciales de Kubernetes déployables dans les data centers d'entreprise. Les infrastructures proposées incluent notamment des serveurs bare-metal, des machines virtuelles fonctionnant sur un hyperviseur et des infrastructures IaaS dans le cloud public.

Vous trouverez ci-dessous une liste des principaux fournisseurs proposant une distribution commerciale de Kubernetes :

Canonical

Canonical Ubuntu est le système d'exploitation Linux le plus recherché sur le cloud public. Ubuntu fonctionne également dans des clouds privés et sur des appareils IoT, de même que comme système d'exploitation PC. Les entreprises choisissent Ubuntu pour sa compatibilité, sa performance et sa polyvalence.

Canonical utilise deux versions de distribution Kubernetes :

- MicroK8s – Cette distribution ultralégère peut s'installer sur des appareils IoT et de périphérie avec une seule commande.
- Charmed Distribution of Kubernetes – Cette version upstream de Kubernetes est conforme à la CNCF. Canonical propose aux entreprises un service managé comprenant la formation, l'installation, la configuration, des outils d'orchestration et l'optimisation de la production des clusters Kubernetes.

Mirantis

Ce « pure player » d'OpenStack est devenu un spécialiste Kubernetes. Mirantis, qui dispose aujourd'hui de sa propre distribution de Kubernetes, se positionne comme une entreprise d'infrastructure de cloud ouvert avec une plateforme capable de gérer en toute transparence des machines virtuelles et des containers.

En 2019, Mirantis a racheté l'activité Docker Enterprise à Docker, Inc. avec notamment Docker

Engine Enterprise Edition, Docker Desktop Enterprise, Docker Trusted Registry et Docker Enterprise CaaS.

Outre Docker Enterprise, Mirantis propose également une offre managée Kubernetes. Baptisée KaaS (Kubernetes as a Service), celle-ci peut être déployée sur OpenStack ou AWS.

Rancher

Rancher Labs (racheté par SUSE en juillet 2020) offre aux entreprises une plateforme de déploiement et de gestion Kubernetes.

Rancher Kubernetes Engine cible les départements IT des entreprises qui envisagent de fournir Kubernetes as a Service à leurs salariés. En plus de maintenir sa propre distribution de Kubernetes, Rancher peut également gérer les offres CaaS – Amazon Elastic Container Service for Kubernetes (Amazon EKS), Azure Kubernetes Service (AKS), Google Kubernetes Engine (GKE), etc. – tout en faisant appliquer des politiques de sécurité homogènes. L'intégration de cette plateforme avec Active Directory, LDAP et d'autres services IT permet d'assurer l'authentification, le contrôle d'accès basé sur les rôles et la conformité aux politiques de sécurité sur de multiples clusters Kubernetes.

Red Hat

Red Hat OpenShift Container Platform est une plateforme applicative reposant sur Kubernetes. Elle s'appuie sur des technologies open source éprouvées telles que Red Hat Enterprise Linux, CRI-O et Kubernetes pour assurer une gestion du cycle de vie applicatif de bout en bout.

Le déploiement de Red Hat OpenShift Container Platform peut s'effectuer dans le cloud public ou sur site. La plateforme automatise les builds, les déploiements, l'évolutivité et la gestion de l'intégrité des containers et des applications. En tant que distribution Kubernetes certifiée par la CNCF, OpenShift garantit la portabilité et l'interopérabilité des workloads containerisés.

Le déploiement d'OpenShift peut se dérouler dans les environnements de cloud privé et de cloud hybride, mais aussi dans le cloud public sous forme de service managé. Red Hat collabore avec des fournisseurs de services tels qu'Amazon, Microsoft et Google pour offrir aux clients une PaaS OpenShift managée.

SUSE

SUSE, la célèbre entreprise de distribution de Linux, a investi le marché des CaaS en proposant une solution Kubernetes.

La plateforme CaaS de SUSE permet de déployer Kubernetes avec de multiples nœuds master dans les environnements cloud publics et privés. Elle intègre des fonctionnalités de sécurité telles que le contrôle d'accès basé sur les rôles, l'analyse des images et le support TLS. La plateforme dispose également d'un système d'exploitation hôte de containers, d'un runtime de containers et d'un registre d'images de containers.

SUSE utilise sa présence au sein des entreprises pour promouvoir la plateforme CaaS. À l'instar de ses concurrents, SUSE s'oriente vers la fourniture d'un environnement intégré basé sur les technologies open source les plus avancées.

VMware

Après avoir acquis Pivotal Labs, VMware a décidé de commercialiser Pivotal Kubernetes Service (PKS) sous la marque VMware Tanzu Kubernetes Grid.

La solution Kubernetes de VMware est disponible en deux versions : Tanzu Kubernetes Grid et Tanzu Kubernetes Grid Integrated Edition (TKGI).

Tanzu Kubernetes Grid est un runtime Kubernetes certifié CNCF conçu pour rationaliser les opérations sur une infrastructure multicloud. Son déploiement peut s'effectuer sur vSphere (sur site), une IaaS (cloud public) ou des appareils aux ressources limitées (périphérie/IoT).

Tanzu Kubernetes Grid Integrated Edition (TKGI) est une solution de containers pour les environnements Kubernetes en production dotée de fonctionnalités réseau avancées, d'un registre de containers privé et d'un système de gestion du cycle de vie complet. Cette solution étroitement intégrée à vSphere, vCenter, vSAN et NSX peut être déployée dans un data center exécutant un environnement VMware ou dans un cloud public exécutant VMware Cloud (VMC) Foundation.

Plateforme managée de gestion de containers

Vous trouverez ci-dessous une liste des principaux fournisseurs proposant Kubernetes sous forme de service managé :

Amazon Elastic Kubernetes Service

Amazon Elastic Kubernetes Service est une offre managée Kubernetes d'AWS. EKS s'appuie sur les composants clés d'AWS que sont Amazon Elastic Compute Cloud (EC2), Amazon EBS, Amazon Virtual Private Cloud (VPC) et Identity Access Management (IAM). Avec son registre de containers intégré appelé Amazon Elastic Container Registry (ECR), AWS permet d'accéder aux images de containers avec une faible latence et de manière sécurisée.

Amazon EKS offre une parfaite intégration à CloudWatch pour la surveillance et à un système IAM pour la sécurité. AWS App Mesh est muni de fonctionnalités natives de maillage de services conçues pour les workloads déployés dans AWS. Avec AWS Fargate, Amazon a lancé un moteur de calcul sans serveur pour l'exécution des containers dans EKS.

Pour accélérer l'apprentissage et l'inférence du machine learning, il est possible de programmer les workloads sur des nœuds attachés aux processus graphiques (GPU) NVIDIA.

AWS Outposts, la plateforme de cloud hybride d'Amazon, exécute EKS en interne.

Azure Kubernetes Service

La plateforme managée de gestion de containers Azure Kubernetes Service (AKS) fonctionne avec Microsoft Azure. AKS s'intègre aux VM Azure, à Azure Storage, Virtual Networking et Azure Monitoring. Vous pouvez provisionner Azure Container Registry (ACR) dans le même groupe de ressources que le cluster AKS pour disposer d'un accès privé aux images de containers.

AKS s'appuie sur des groupes de machines virtuelles (Virtual Machine Scale Sets) et des groupes de disponibilité (Availability Sets) pour augmenter ou diminuer le nombre de nœuds dans un cluster.

AKS est disponible dans Azure Stack Hub, la solution de cloud hybride de Microsoft conçue pour exécuter les services Azure dans les data centers.

Google Kubernetes Engine

Google Kubernetes Engine (GKE) a été l'un des tous premiers services managés à offrir Kubernetes dans le cloud public. Comme d'autres CaaS dans le cloud public, GKE utilise les services clés de Google Cloud Platform (Compute Engine, Persistent Disks, VPC, Stackdriver, etc.).

Google propose Kubernetes dans les environnements sur site, ainsi que sur d'autres plateformes de cloud public via le service Anthos. Bien qu'exécuté sous GCP, ce plan de contrôle gère le cycle de vie des clusters lancés dans les environnements hybrides et multicloud.

Google a intégré à GKE un service managé Istio conçu pour offrir des fonctionnalités de maillage de services. Sa plateforme sans serveur Cloud Run, basée sur le projet open source Knative, exécute quant à elle les containers sans lancer les clusters.

IBM Cloud Kubernetes Service

IBM Cloud Kubernetes Service (IKS) est une offre managée conçue pour créer des clusters Kubernetes de serveurs hôtes. Elle permet de déployer et de gérer dans IBM Cloud des applications containerisées. Cette solution certifiée Kubernetes fournit une planification intelligente, une auto-réparation, une évolutivité horizontale, une découverte de services et un équilibrage des charge, des rollouts et rollbacks automatisés, ainsi qu'une gestion des secrets et de la configuration adaptée aux nouvelles applications.

IBM est l'un des rares fournisseurs cloud à offrir un service managé Kubernetes en configuration bare-metal.

Depuis son rachat de Red Hat, l'entreprise peut proposer un choix de clusters Kubernetes ou OpenShift disponibles via IKS.

IBM a par ailleurs développé Cloud Paks, un ensemble d'appliances hyperconvergées basées sur Kubernetes et OpenShift exécutables dans les data centers d'entreprise.

Autres services managés Kubernetes

Parmi les autres services managés Kubernetes, on retrouve Alibaba Cloud Container Service, DigitalOcean Kubernetes, Huawei Cloud Container Engine et Platform9.

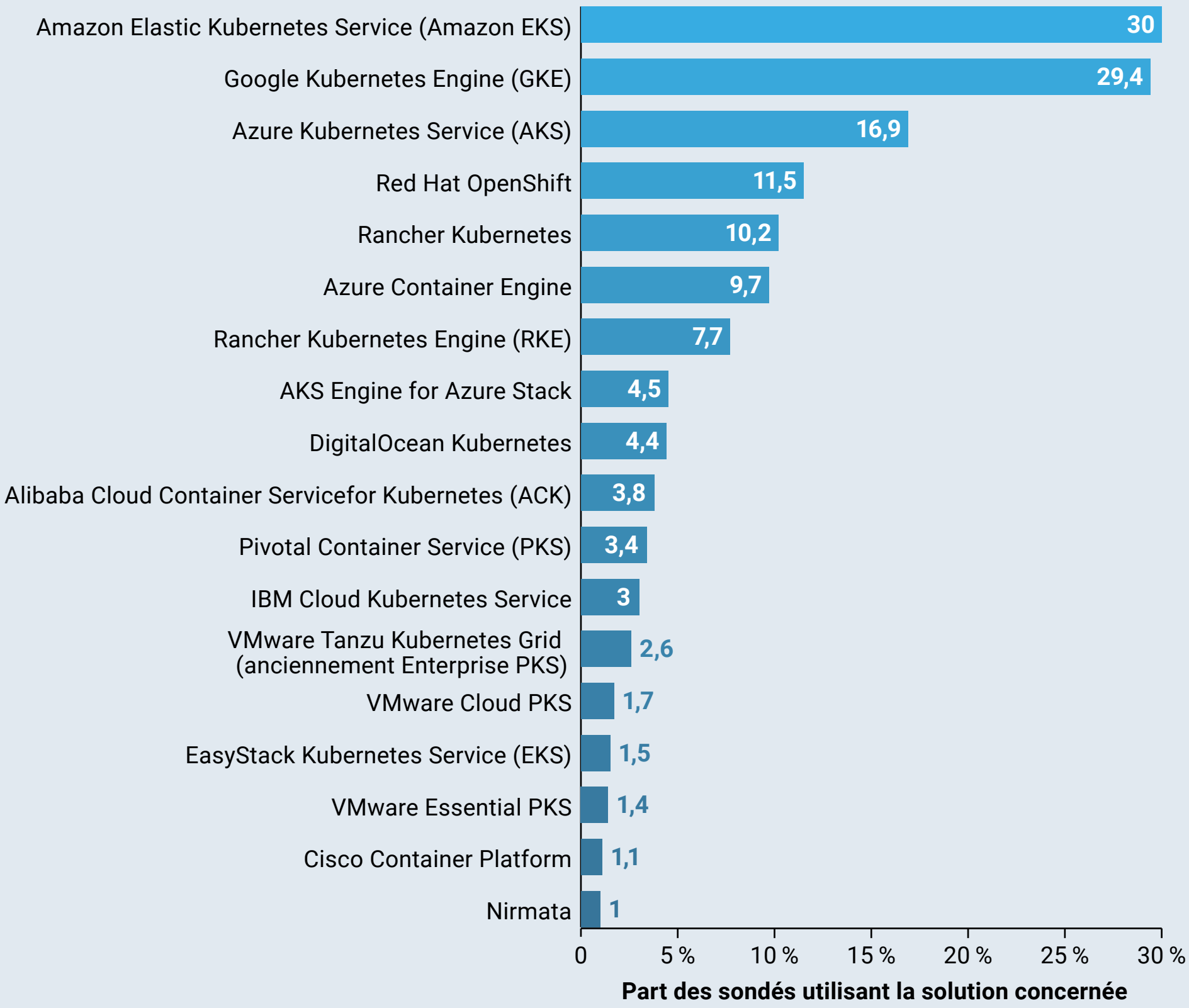


Figure 2.7 : *Les noms de plusieurs solutions ont changé depuis ce sondage. En y intégrant les résultats de Pivotal et de VMware, Tanzu se sera probablement hissé dans le classement de notre étude 2020.*

Adoption des offres Kubernetes de fournisseurs cloud

Lors de la réalisation de l'étude CNCF de 2017, 25 % des utilisateurs Kubernetes sondés exécutaient leurs containers avec Google Kubernetes Engine (GKE). À cette époque, AWS mettait encore en avant sa solution Elastic Container Service (ECS), tandis que Microsoft venait tout juste de lancer Azure Container Service. Pourtant, Kubernetes était déjà sur le

point de s'imposer comme l'orchestrateur de containers de référence. Bien que l'offre Kubernetes n'en était encore qu'à ses débuts, 52 % des utilisateurs Kubernetes exécutaient ce système dans un environnement AWS, 34 % sur Google Cloud Platform (GCP), 21 % sur Azure et 15 % sur VMware. La moitié des workloads Kubernetes tournaient sur AWS, même s'ils étaient exécutés sur des machines virtuelles et gérés par les clients AWS.

Cette année-là, AWS annonçait lors de son re:Invent le lancement d'Amazon Elastic Container Service for Kubernetes (EKS) — avec une mise en service pour mi-2018. Dans la foulée, AWS rejoignait la CNCF. Si Kubernetes venait de gagner la bataille de l'orchestration de containers, les entreprises n'en avaient pas moins tendance à exécuter leurs containers là où elles faisaient tourner leurs autres workloads. Cette situation a entraîné une hausse significative de l'adoption d'Amazon EKS et d'Azure Kubernetes Service (AKS) en 2019.

La dernière étude CNCF, publiée en 2019, proposait plus de 100 réponses pour sa question portant sur la gestion des containers. Amazon EKS se classait en tête avec 30 % des utilisateurs Kubernetes, juste devant GKE (29 %). Outre Azure Kubernetes Service (AKS), Red Hat OpenShift et Rancher Kubernetes affichaient eux aussi des scores d'adoption plus qu'honorables.

Kubernetes a évolué. Qu'en est-il de votre sécurité ?

Kubernetes a beau intégrer un grand nombre de fonctionnalités de sécurité, il n'est pas forcément protégé par défaut. Il faut encore sécuriser la gestion des dépendances, sans oublier que de nouveaux vecteurs d'attaque (containers malveillants, par exemple) font leur apparition. Malgré les progrès en matière de sécurité, l'API reste le principal point d'accès à un système Kubernetes pour les attaquants.

La bonne nouvelle c'est que ces dernières années, les équipes de sécurité en ont appris beaucoup plus sur la manière de protéger les déploiements Kubernetes et les applications s'exécutant sur des containers. Ces menaces peuvent être traitées plus facilement grâce à une combinaison de workflows et d'outils utilisés par les développeurs, les équipes de sécurité et les opérations IT (DevSecOps). Par exemple, des outils de détection et d'analyse des anomalies permettent de repérer en toute simplicité les containers malveillants et autres vecteurs d'attaque.

Évangéliste de la sécurité cloud pour Palo Alto Networks Prisma Cloud, Robert Haynes nous parle de la sécurité de Kubernetes au-delà de ses fonctionnalités natives. Il revient également sur l'évolution des vulnérabilités de Kubernetes depuis les premières attaques d'API survenues il y a quelques années. Ces premières attaques sérieuses « nous ont rappelé ce que nous savions déjà… Kubernetes est un logiciel et tous les logiciels comportent des vulnérabilités », explique Robert Haynes.

Écouter sur SoundCloud

Robert Haynes *affiche une carrière qui s'étend du helpdesk à l'administration des systèmes Unix, en passant par le marketing. Il fait aujourd'hui partie de l'équipe Palo Alto Networks Prisma Cloud, où il aime échanger sur des sujets techniques concernant la sécurité, les technologies et les individus.*

Comment éliminer la fatigue liée à la complexité de Kubernetes

Plus les entreprises se tournent vers Kubernetes et les environnements cloud-native à grande échelle, plus la gestion des clusters se complexifie. La « fatigue liée à la complexité de Kubernetes » est devenue une véritable préoccupation.

Dans ce podcast, Ravi Lachhman, évangéliste chez Harness, et Frank Moley, responsable technique senior chez DataStax, se penchent sur ce problème et sur les solutions possibles.

Une grande partie de cette fatigue peut être attribuée à la façon dont les équipes DevOps doivent étendre leurs domaines d'expertise, souvent au-delà de leur zone de confort, pendant la migration. Par exemple, l'équipe d'infrastructure doit être davantage en phase avec les besoins des développeurs, tandis que les workloads de développement couvrent de plus en plus de tâches associées à l'infrastructure.

Aider vos équipes DevOps à éviter la fatigue liée à la complexité de Kubernetes passe par une prise de conscience de vos limites. Achetez les composants au fur et à mesure, conseille Ravi Lachhman, et apprenez à mieux gérer les écarts afin d'atteindre vos objectifs.

Écouter sur SoundCloud

Ravi Lachhman *est évangéliste chez Harness. Avant de rejoindre Harness, il a été évangéliste chez AppDynamics et a occupé divers postes de vente et d'ingénierie chez Mesosphere, Red Hat et IBM.*

Frank Moley *est responsable technique senior chez DataStax et rédacteur de contenus pour LinkedIn Learning. Il se concentre sur l'informatique cloud-native, les architectures de microservices, l'ingénierie de plateformes et la sécurité.*

Kelsey Hightower sur son parcours personnel avec Kubernetes

Dans ce podcast, nous nous entretenons avec Kelsey Hightower, figure éminente de l'écosystème Kubernetes, Principal Developer Advocate chez Google Cloud et ancien membre du comité technique chargé de superviser tous les projets de la Cloud Native Computing Foundation (CNCF), y compris Kubernetes. Kelsey Hightower y parle de son rôle dans le développement de Kubernetes et des défis à venir.

« Je me souviens avoir parcouru le codebase et qu'il n'y avait aucun avis intéressant même sur la façon de l'installer », évoque-t-il de son premier contact avec Kubernetes en 2015. « La première chose que j'ai faite a été de créer ce que certains considèreraient comme l'un des tout premiers guides d'installation. »

Aujourd'hui, Kelsey Hightower souhaite voir la courbe d'apprentissage baisser à mesure que la plateforme se développe. « Les développeurs ne devraient pas avoir à devenir des experts des réseaux Kubernetes pour faire leur travail d'ingénieurs logiciels », ajoute-t-il.

Écouter sur SoundCloud

Kelsey Hightower est Principal Developer Advocate chez Google Cloud. Il a contribué au développement et au perfectionnement de nombreux produits Google Cloud (Google Kubernetes Engine, Google Cloud Functions, Apigee API Gateway, etc.).

Conclusion

Lancé il y a plus de cinq ans, Kubernetes est aujourd'hui relativement complet. Désormais, les concepts de pods, de nœuds et de clusters font partie intégrante de la terminologie des architectures scale-out. Il ne s'agit plus de s'émerveiller devant cette technologie, mais bel et bien de développer des composants sur Kubernetes.

Si Kubernetes n'enflamme plus les foules, c'est parce qu'en 2020 il est résilient et évolutif. Il joue parfaitement son rôle de socle.

Il est facile d'oublier que Kubernetes était tout sauf conventionnel il y a quelques années. Depuis 2017, année de la publication du premier eBook de The New Stack consacré à cette technologie, un processus itératif de mises à jour continues de l'orchestrateur de containers a permis de consolider l'architecture sous-jacente des applications scale-out distribuées.

Reste à savoir comment développer des composants sur Kubernetes. Dans l'univers des architectures scale-out, le code est réparti à travers des containers. En d'autres termes, le code est le composant du container. Ce dernier est un processus exécuté sur un système d'exploitation. En tant que processus, le container est portable. Conçu pour être immuable, il peut être remplacé par un nouveau container afin de mettre à jour le code du composant applicatif. L'infrastructure immuable repose sur l'idée selon laquelle il est possible de déployer une application dans un environnement homogène et donc de la mettre à jour continuellement, en remplaçant tout simplement les containers par d'autres renfermant le code le plus récent.

Or, c'est parce que les containers sont immuables que le développement sur Kubernetes n'est pas toujours une mince affaire. Les containers sont sans état. Pour que Kubernetes soit utile aux entreprises, les ingénieurs ont dû développer des interfaces pour connecter ces containers.

L'interface réseau pour containers (CNI) relie les containers et les plug-ins associés au réseau utilisé par l'organisation pour exécuter une application. Dans un tel environnement, une vaste prise en charge s'avère essentielle pour permettre aux plug-ins de se connecter aux réseaux. Pour comprendre les raisons du succès de la communauté Kubernetes, il suffit de s'intéresser aux entreprises qui développent une CNI commune. Les projets Kubernetes font la part belle

à la coopération dans un cadre de développement communautaire ouvert. Si la mécanique Kubernetes s'est désormais banalisée, les techniciens chargés de développer l'architecture sous-jacente sont des professionnels déterminés et passionnés. Ils signent ainsi l'architecture d'un des premiers systèmes d'exploitation pour logiciels et services exécutés sur une infrastructure distribuée avancée.

Mais Kubernetes reste une technologie relativement récente. Les systèmes d'exploitation client sont destinés aux ordinateurs personnels et aux serveurs, essentiellement pour des applications monolithiques. Linux domine l'univers des serveurs. C'est sur Linux que reposent les containers – en tant qu'extensions du noyau Linux. De son côté, Kubernetes fait du container un processus de base de l'orchestration du code exécuté comme composant. Il offre un plan de contrôle indépendant des architectures cloud et client. Il permet aux API de gérer le plan de contrôle et les ressources serveur, réseau et de stockage correspondantes.

Le caractère récent de Kubernetes a un impact sur l'expérience des développeurs. En effet, il reste encore à optimiser l'architecture Kubernetes pour ces professionnels. Beaucoup moins pertinentes qu'il y a trois ans, les plateformes sous forme de service (PaaS) ont fait leur temps. Une PaaS sert à exécuter des applications sur des services cloud et des environnements d'entreprise. Elle est développée et assemblée comme une structure applicative. Seulement voilà, même les meilleures PaaS ne peuvent prendre en charge certains cas particuliers, surtout les applications monolithiques d'ancienne génération.

Les containers changent la donne. Le code exécuté dans un container est bien plus facile et bien plus rapide à gérer. Toutefois, les développeurs disposent encore de peu de méthodes pour développer des services sur Kubernetes. C'est là que les containers sous forme de service (CaaS), ou plateformes de gestion de containers entrent en jeu. Le CaaS intègre différents services cloud-native pour les exécuter tant sur des architectures d'entreprise monolithiques que sur des architectures modernes de microservices. Une plateforme de gestion exécute de multiples services sur Kubernetes. À commencer par le processeur et l'infrastructure physique. Elle comprend entre autres le système d'exploitation optimisé et le runtime des containers, la réseautique container-native, le stockage, la sécurité, le registre, l'observabilité et le moteur d'orchestration des containers.

La maturation des plateformes CaaS prendra du temps. Les services managés gagneront encore du terrain, ce qui simplifiera la tâche des développeurs.

En attendant, les développeurs, les professionnels responsables de la configuration des services et ceux chargés de définir les politiques Kubernetes verront leur rôle évoluer. L'univers des containers renouvelle le concept même du système d'exploitation, où qu'il s'exécute sur l'infrastructure.

Au bout du compte, Kubernetes n'est peut-être pas si banal. Mais il le deviendra tôt ou tard. Comme Kelsey Hightower aime à le dire, la prochaine étape consiste à le rendre imperceptible.

Avertissement

En plus de nos sponsors, cet eBook mentionne les entreprises suivantes, qui sont des sponsors de The New Stack :

Accurics, AppDynamics, Aspen Mesh, Amazon Web Services, CircleCI, Citrix, CloudBees, Cloud Foundry Foundation, Cockroach Labs, Dell Technologies, Diamanti, Futurewei, GitLab, HAProxy, HashiCorp, Honeycomb, InfluxData, Lightbend, Lightstep, LogDNA, MayaData, MongoDB, New Relic, NS1, Packet, PagerDuty, Portworx, Red Hat, Redis Labs, Rezilion, SaltStack, SAP, Sentry, Snyk, Sonatype, The Linux Foundation, TriggerMesh et VMware.