Tendances technologiques

Conteneurs d'applications
Version 0.1
Date 2019-05-28

Shared Services Canada
Services partagés Canada
Canada

Table des matières

Sommaire opérationnel

Les conteneurs d'applications sont un type de virtualisation des SE, en ce sens que les applications distribuées sont exécutées sur une seule machine virtuelle ou hôte matériel plutôt que sur des machines virtuelles ou hôtes distincts. Un SE hôte peut prendre en charge plus d'un conteneur, et on peut exécuter ces derniers sur des systèmes divers, notamment de façon délocalisée ou sur des machines virtuelles, et sur des SE différents comme Linux, Windows, et Macintosh. Les conteneurs d'applications présentent d'immenses avantages, car on peut les déployer en environnement infonuagique. Cependant, leur avantage prépondérant, c'est leur architecture : comme on peut les exécuter à partir de machines virtuelles ou matérielles différentes, les applications ainsi déployées sont plus facilement accessibles. Ces machines peuvent faire partie d'une architecture infonuagique ou non, et cela donne une grande souplesse à la gestion de la charge de travail, et permet aux développeurs de créer des systèmes qui tolèrent les pannes.

Les conteneurs d'applications recentrent les centres de données : d'une optique axée sur les machines, ils doivent passer à une optique fondée sur les applications. Comme les conteneurs englobent l'environnement de l'application, et abstraient la machine, le système d'exploitation, le développeur et l'infrastructure de déploiement. Les conteneurs et les images de conteneurs bien conçus se cantonnent à une seule application, gérer les conteneurs revient à gérer des applications plutôt que des machines. La transition des interfaces applicatives, auparavant axées sur les machines, à une orientation axée sur les applications, simplifie considérablement le développement des applications.

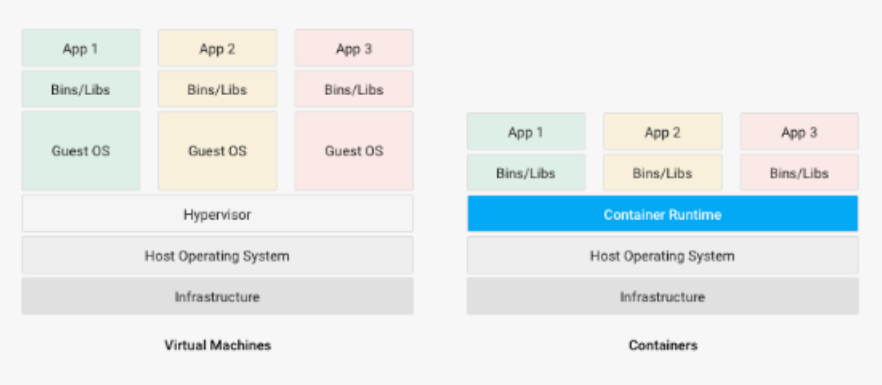

Figure 1: Architecture comparée
des machines virtuelles et des conteneurs d'applications

Exposé technique

Pour penser clairement à propos des conteneurs et leur fonctionnement, il est utile de les voir comme des images système. On peut agencer ces images de façon hiérarchique, et lier des applications à cette hiérarchie. Mieux, chaque nœud de cette hiérarchie en arborescence permet aux utilisateurs de partager des images entre applications. On peut ainsi configurer des états binaires; en d'autres termes il n'est pas nécessaire de déplacer la pile complète de l'application dans un seul fichier. Le lien entre images et conteneurs ressemble à celui entre classe et objet, en programmation orientée objet : tout objet est une instance d'une classe. Une *pomme*, par exemple, est une instance de la classe *fruit*. La classe dicte les données enregistrées sur l'objet et les fonctions qu'on peut appliquer è l'objet. Dans le cas des conteneurs d'applications, l'image, tout comme la classe, constitue le modèle appliqué au conteneur. En d'autres termes, le conteneur ressemble à une instance (ou objet) de cette classe.

Cette architecture présente un autre avantage : en cas de découverte d'une vulnérabilité dans un nœud précis, l'arborescence permet de vérifier tous les autres nœuds pouvant être touchés. Avant de créer des images, il importe d'étudier un exemple. Docker est un système de création de conteneurs; il se sert d'un fichier Docker pour créer les images. Les conteneurs servent à isoler les processus exécutés dans le SE.

On peut considérer les conteneurs d'applications comme une forme de virtualisation semblable aux machines virtuelles (MV). À la différence de ces dernières, cependant, les conteneurs d'applications virtualisent le niveau système d'exploitation plutôt que la pile complète [1]. C'est pourquoi ils sont bien plus légers, et ne nécessitent qu'une fraction de la mémoire et de la puissance de traitement des MV, car un seul SE peut héberger de nombreux conteneurs.

Les conteneurs d'applications simplifient la mise en place d'une architecture axée sur les services (AAS). Dans ce type d'architecture, une application est atomisée en ses services constitutifs. Or, les conteneurs permettent de lancer chaque service dans un conteneur distinct. La mise en grappe permet ensuite de mettre à l'échelle les services au besoin. Cependant, il faut les outils d'orchestration appropriés pour synchroniser le tout et permettre la mise à l'échelle. Ces outils d'orchestration des conteneurs gèrent le cycle de vie des conteneurs; une équipe de développement pourra aussi s'en servir pour créer et déployer les conteneurs, vérifier la redondance et la disponibilité, et transférer les conteneurs d'un hôte à l'autre.

On peut aussi déployer en conteneur des applications monolithiques : comme ce sont essentiellement des applications à un niveau; une instance sera déployée dans un conteneur. Cela diffère de la stratégie AAS ou microservices, car un conteneur représente un service de cette application. Avec les conteneurs, mettre à l'échelle une application monolithique veut dire déployer de multiples instances du conteneur, et donc autant de conteneurs. Docker permet de déployer les mises à jour d'une

application dans des images Docker, une méthode rapide et économe en ressources réseau. Et, comme on peut multiplier les conteneurs plus rapidement, créer d'autres instances d'une application est aussi plus rapide.

Docker et CoreOS sont les deux grands fabricants de technologies mettant en œuvre les conteneurs. Comme Docker a été le premier, c'est le plus utilisé par l'industrie. Les conteneurs d'applications sont les plus utiles aux entreprises d'envergure. Docker est sur le marché depuis 2014; selon un sondage de Datadog mené en 2016, l'adoption de Docker a augmenté de plus de 30 pour cent par rapport à 2015. Point important sur ces statistiques : les entreprises sondées exploitent Docker en production plutôt qu'uniquement pour le développement, ce qui laisse penser qu'elles utilisent les conteneurs dans une stratégie à long terme plutôt que simplement en faire l'essai.

On peut voir Docker comme le « moteur » de gestion et de création des conteneurs, mais ce n'est pas le seul. D'autres, comme OpenShift, Kubernetes et Rancher, proposent divers outils pour gérer les conteneurs.

OpenShift gère les conteneurs à l'aide de Docker et de Kubernetes en arrière-plan : ce logiciel s'intègre à Kubernetes pour automatiser des fonctions comme l'échelonnage, le déploiement et la gestion de l'état de santé des conteneurs. Originellement créé par Google, Kubernetes est un outil d'orchestration bien conçu servant à gérer les conteneurs. Il présente au client une interface applicative de haut niveau avec laquelle on peut regrouper les conteneurs logiquement, gérer les charges et créer des ensembles de conteneurs. Rancher, quant à lui, est un outil de gestion des conteneurs créé sur la plateforme de conteneurs Kubernetes.

Utilisation par l'industrie

Plusieurs organisations d'envergure ont souligné les avantages d'intégrer les conteneurs à leur cycle de développement. ADP, par exemple, crée des solutions infonuagiques de gestion des ressources humaines; elle compte 630 000 clients et grosso modo 35 millions d'utilisateurs de ses systèmes. Elle a choisi d'accélérer son cycle de développement à l'aide de conteneurs d'applications : en avril 2017, elle exploitait 469 moteurs Docker et 3 771 conteneurs roulant sur des machines virtuelles. PayPal, qui héberge la plus grande plateforme de paiement en ligne au monde, est aussi friand des conteneurs d'applications. Elle compte plus de 210 millions d'utilisateurs dans 200 marchés à l'étranger, et traite chaque trimestre des transactions d'une valeur de près de 100 milliards de dollars. PayPal a modernisé ses centres de données à l'aide de Docker, et, en 2016, l'entreprise a déployé son premier conteneur en AQ et en production. Comme ses applications étaient maintenant indépendantes du système d'exploitation sous-jacent, PayPal a pu moderniser son infrastructure sans devoir mettre à jour ses applications. Avec une infrastructure plus moderne et plus rapide, PayPal a observé une amélioration des performances de 10 à 20 pour cent, sans la moindre interruption de service. L'entreprise a maintenant migré 700 applications vers Docker, et compte 45 000 MV en production qui hébergent plus de 150 000 conteneurs.

Utilisation par le gouvernement du Canada

Peu d'initiatives ou de programmes connus du gouvernement du Canada (GC) encouragent l'utilisation actuelle ou future des conteneurs d'applications et des technologies connexes. Les conteneurs d'applications, en tant qu'élément stratégique en TI du GC, sont absents tant du Plan stratégique des opérations numériques du GC de 2018 à 2022 que du Plan stratégique du GC pour la gestion de l'information et la technologie de l'information de 2017 à 2021. Cela peut découler des priorités actuelles de SPC, c'est-à-dire le regroupement et la transformation numérique de concert avec la mise en œuvre des services infonuagiques. L'essentiel des ressources et des efforts est consacré à résoudre les problèmes de mise en œuvre et à régler des questions entourant la protection des données confidentielles des Canadiens.

L'arrivée sur le marché des conteneurs démontre toutefois que les organisations d'envergure qui exploitent les réseaux et travaillent au développement d'applications infonuagiques peuvent bénéficier considérablement des conteneurs d'applications.[i] Il est certes possible de mettre en œuvre les applications d'infrastructure qui assurent les services infonuagiques exclusivement sur des machines virtuelles, mais les frais de maintenance découlant de l'utilisation de plusieurs systèmes d'exploitation différents sur des MV distinctes en annulent les avantages.[ii] Or, les conteneurs et les technologies connexes constituent une architecture de remplacement des MV sinon complémentaire à celles-ci. Avec la transition du GC vers les services infonuagiques et le développement d'applications infonuagiques, exploiter les conteneurs d'applications et en orchestrer le fonctionnement par Kubernetes pourra faire partie intégrante de l'architecture de TI du GC.

Répercussions pour Services partagés Canada

Proposition de valeur

Les conteneurs d'applications ont une multitude d'utilisations possibles, et SPC peut certes tirer parti d'un outil si souple. Les conteneurs d'applications sont particulièrement utiles pour le développement et la mise à l'essai de nouvelles applications, car les déployer pour créer un environnement sûr pour l'exécution de code, comme un bac à sable, est un jeu d'enfant. Un fichier de configuration permet de configurer facilement chaque conteneur, notamment l'échelonner pour répondre à tout besoin en ressources. Comme leur configuration est très simple, les conteneurs d'applications sont souvent intégrés au cadre de développement des logiciels destinés à une architecture axée sur les services (AAS), pour héberger des microservices. Une application développée pour une AAS consiste souvent en de nombreux petits éléments (les microservices) qui fonctionnent ensemble et s'échangent des données.[iii] Si SPC adopte l'AAS ou un autre modèle de développement semblable, comme l'architecture

fondée sur les microservices, les conteneurs d'applications seront un outil de développement incontournable.

Les conteneurs peuvent simplifier et accélérer le déploiement des applications. On peut même prévoir des économies supplémentaires, car les besoins en mémoire du serveur hôte sont plus modestes si on utilise les conteneurs. On peut aussi créer plus d'un conteneur pour une même application, comme dans l'AAS. En pratique, cela simplifie le contrôle des versions : plutôt qu'appliquer un correctif directement, on peut plutôt retirer le conteneur qui héberge un élément de l'application puis le mettre à jour sans perturber le fonctionnement de l'application. Si on gère tous les conteneurs d'une application à l'aide d'un orchestrateur, le risque de perturbation est encore plus bas, car cet outil peut surveiller l'état de santé de tous les conteneurs qu'il gère.[iv] S'il détecte qu'un conteneur ne fonctionne pas comme prévu, il peut le mettre hors service automatiquement et en lancer d'autres pour compenser.

De plus, les conteneurs sont portables : ils exécutent les programmes qu'ils contiennent de façon identique à chaque lancement, où qu'ils soient.[v] En d'autres termes, les applications sont exécutées de façon uniforme en environnement normalisé, qu'on lance le conteneur à partir du nuage ou d'un serveur matériel. Cela élimine à toutes fins utiles les installations et configurations fastidieuses sur chaque ordinateur d'un réseau : il suffit plutôt de lancer l'application à partir d'un conteneur. Cette portabilité peut aussi simplifier l'utilisation d'applications désuètes; si par exemple une application a besoin d'un environnement précis, comme des versions précédentes de Java ou Python, il suffit de recréer cet environnement dans un conteneur. Des restrictions restent toutefois présentes : il est impossible de lancer les conteneurs créés sur une plateforme, comme Docker, sur une autre plateforme comme Linux Containers, il certains problèmes de compatibilité en amont persistent, car les conteneurs créés pour une version récente d'une plateforme peuvent ne pas fonctionner sur une version plus ancienne de cette même plateforme.[vi]

Enjeux

Quatre obstacles de taille risquent d'empêcher l'adoption des conteneurs d'applications :

Éducation et formation du personnel – Pour bien exploiter les conteneurs, il faut de toutes nouvelles compétences. Les équipes de TI doivent aussi bien connaître la plateforme de conteneurs et l'outil de gestion des conteneurs qu'ils utiliseront. En outre, ces équipes devront peut-être apprendre de nouvelles balises de conception, comme l'AAS, avant de pouvoir tirer pleinement parti des conteneurs d'applications.

Choix de la bonne plateforme et du bon outil d'orchestration – Cette décision cruciale dépend des besoins de chaque organisation. Divers outils sont disponibles, en fonction du système d'exploitation sur lequel rouleront les conteneurs d'application. Le GC

envisage l'adoption d'outils de source ouverte; les options abondent et chacune a ses points forts et points faibles.

Reconception des applications – Les applications existantes développées selon un modèle monolithique ne se prêtent pas toutes à l'environnement des conteneurs. Des problèmes peuvent survenir si les développeurs mettent à jour une fonction d'une application existante, mais cela vaut essentiellement pour toutes les applications monolithiques. On peut toutefois les scinder en éléments plus petits qui eux peuvent fonctionner en conteneur, ce qui permet de les utiliser quand même sur les systèmes plus récents.

Sécurité – La sécurité est vitale, car on peut déployer les conteneurs en mode infonuagique, et en parallèle dans un réseau local. Question névralgique : comment concevoir une solution performante qui libère les utilisateurs sans être vulnérable aux fuites de données.

Considérations

En règle générale, les conteneurs ont des limites, comme toute technologie. Il est donc inutile de les adopter simplement pour innover. Les conteneurs d'applications comportent leurs propres difficultés et exigences, notamment le développement de compétences et le choix de l'ensemble logiciel adéquat. Si SPC s'oriente vers un modèle qui les rend incontournables, l'investissement voulu pour surmonter ces difficultés, et les coûts afférents, en vaudront la chandelle.

Sur le plan sécurité, supposer que les applications exécutées en conteneur sont sécurisées d'office serait irresponsable. Il faut tenir compte des questions de sécurité pendant tout le cycle de vie des applications, conteneurs compris. Si SPC adopte cette nouvelle technologie, d'autres mécanismes de sécurité devront être élaborés en fonction de celle-ci.

Les conteneurs constituent un nouvel outil qui présente à SPC bien des avantages potentiels, à condition d'être bien gérés et bien intégrés au milieu de travail.

Références (en anglais)

1. https://cloud.google.com/containers/
2. https://www.networkcomputing.com/data-centers/5-container-deployment-challenges/1532194322
3. https://rancher.com/
4. https://shadow-soft.com/open-source-container-management-tools/
5. https://www.contino.io/insights/whos-using-docker
6. https://searchitoperations.techtarget.com/definition/application-containerization-app-containerization
7. https://www.anmb.ro/buletinstiintific/buletine/2016_Issue2/FCS/412-414.pdf
8. https://www.gartner.com/smarterwithgartner/6-best-practices-for-creating-a-container-platform-strategy/

[i] CENGN. (2018). CENGN and CloudOps Collaborate to Train Industry on Docker and Kubernetes. [en ligne] Source : https://www.cengn.ca/docker-kubernetes-training-jan18/ [consulté le 13 juillet 2018]

[ii] Heron, P. (2018). Experimenting with containerised infrastructure for GOV.UK - Inside GOV.UK. [en ligne] Insidegovuk.blog.gov.uk Source : https://insidegovuk.blog.gov.uk/2017/09/15/experimenting-with-containerised-infrastructure-for-gov-uk/ [consulté le 6 juillet 2018]

[iii] Abeywardhana, Sajith. (9 février 2019). *SOA vs Microservices with Docker and Kubernetes-1*. Medium. Source : https://medium.com/@sajithswa/soa-vs-microservices-with-docker-and-kubernetes-1-291686200a0f [consulté le 18 juillet 2019]

[iv] Wagner, Bill, et al. (1er juin 2019). *Health Monitoring - .NET Microservices: Architecture for Containerized .NET Applications*. Microsoft Developer Division. Redmond, Washington. Source : https://docs.microsoft.com/en-us/dotnet/standard/microservices-architecture/implement-resilient-applications/monitor-app-health [consulté le 19 juillet 2019]

[v] Tozzi, Chris. (2017). *The pros and cos of container platforms for portability.* Search Microservices. Source : https://searchmicroservices.techtarget.com/tip/The-pros-and-cons-of-container-platforms-for-portability [consulté le 22 juillet 2019]

[vi] Tozzi, Chris, *ibid*.