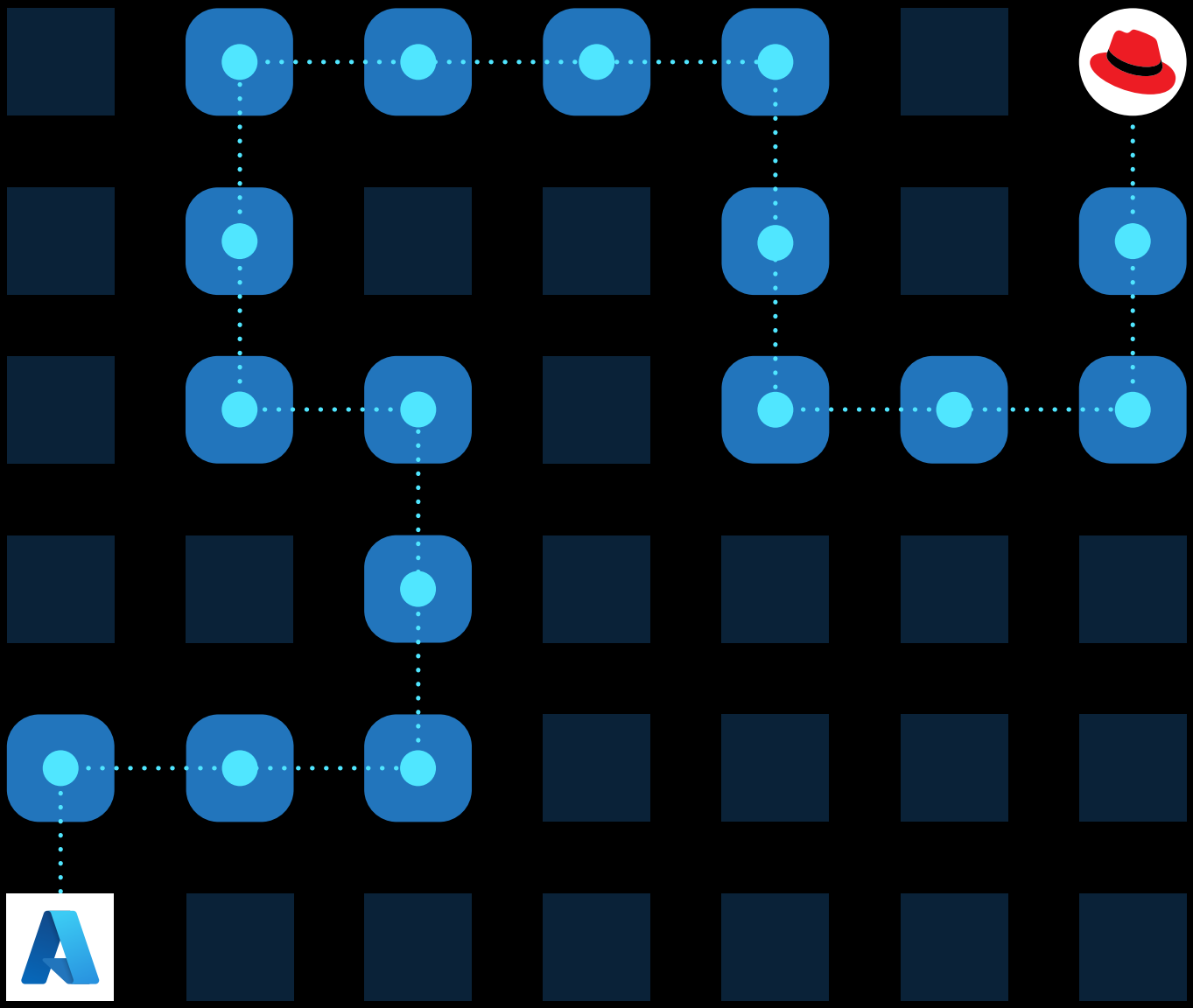
Série de livres numériques

Microsoft Azure

Premiers pas avec Azure Red Hat OpenShift

De la preuve de concept à la production



Premiers pas avec Azure Red Hat OpenShift

Chapitre 1

Communiquer avec Microsoft et Red Hat

Si Azure Red Hat OpenShift vous intéresse, nous aimerions vous contacter et discuter avec vous. Ce manuel fournit des indications d'utilisation utiles de la plateforme. Cependant, Red Hat et Microsoft peuvent fournir de nombreuses ressources supplémentaires, non incluses dans ce manuel, et vous permettre de pousser l'expérience plus loin. Nous souhaitons nous assurer conjointement qu'Azure Red Hat OpenShift est la plateforme d'applications adaptée à vos besoins en matière d'innovation applicative.

La création d'Azure Red Hat OpenShift découle d'une raison simple : des clients comme vous l'ont demandé. Plus que jamais, nos clients communs (entreprises, petites organisations, etc.) déploient le portefeuille de Red Hat sur Microsoft Azure. Si la prise en charge d'OpenShift sur Azure, en tant qu'offre autogérée, est entièrement assurée depuis plusieurs années, la configuration, le déploiement et les opérations du jour 2 concernant la gestion du cluster nécessitent une expertise Kubernetes et du temps. Le temps précieux ainsi perdu, les clients ne peuvent le consacrer aux objectifs de l'entreprise.

De plus en plus de clients réussissent leurs déploiements avec l'offre gérée Azure Red Hat OpenShift. Nous constatons, par ailleurs, qu'ils allouent beaucoup moins de temps à l'installation et aux opérations du jour 2 et en consacrent davantage à leurs applications.

Nous avons créé ce manuel en nous appuyant sur notre expérience pratique, afin de fournir à nos clients les meilleures pratiques pour créer des applications dans Azure Red Hat OpenShift. Nous l'avons rédigé comme un manuel d'auto-assistance. Vous pouvez choisir de lire les chapitres de manière séquentielle, de l'introduction à la conclusion, ou simplement rechercher les informations spécifiques dont vous avez besoin.

Destinataires de ce manuel

Ce manuel s'adresse à un public technique (développeurs, opérateurs et architectes de plateforme) qui cherche à renforcer ses capacités de création et de déploiement d'applications au moyen d'Azure et de Red Hat OpenShift pour le déploiement complet de clusters Red Hat OpenShift entièrement gérés.

Contenu de ce manuel

Dans ce manuel, nous vous accompagnerons à travers les sujets clés pour comprendre et adopter Azure Red Hat OpenShift, depuis la preuve de concept au sein de votre organisation jusqu'au déploiement en production.

Chapitre 2 *Présentation de Red Hat OpenShift :* commence par présenter Red Hat OpenShift, et les raisons pour lesquelles les développeurs, les opérateurs et les architectes de plateforme sont nombreux à choisir cette plateforme d'applications qui s'avère fort utile. Vous découvrirez ensuite pourquoi Red Hat OpenShift est une plateforme privilégiée.

Chapitre 3 *Azure Red Hat OpenShift :* présente le service géré et la manière dont vous en profitez en tant que client.

Chapitre 4 *Pré-provisionnement – Questions sur l'architecture d'entreprise :* couvre les points importants à prendre en compte avant de déployer Azure Red Hat OpenShift et comprend de nombreux conseils sur les meilleures pratiques que de nombreux clients nous ont fait découvrir après les avoir appliqués.

Chapitre 5 *Provisionnement d'un cluster Azure Red Hat OpenShift :* regroupe les ressources clés nécessaires au déploiement d'un cluster Azure Red Hat OpenShift dans la documentation officielle.

Chapitre 6 *Post-provisionnement – Jour 2 :* couvre les tâches de post-provisionnement, souvent appelées « jour 2 ». Ce manuel tente, dans la mesure du possible, d'expliquer comment le service géré, Azure Red Hat OpenShift, vous facilite la tâche par rapport à ce que vous auriez dû faire vous-même.

Chapitre 7 *Déploiement d'applications sur Red Hat OpenShift :* présente succinctement le déploiement d'applications sur la plateforme. Veuillez noter l'absence de différence au niveau de l'exécution de Red Hat OpenShift quel que soit l'environnement.

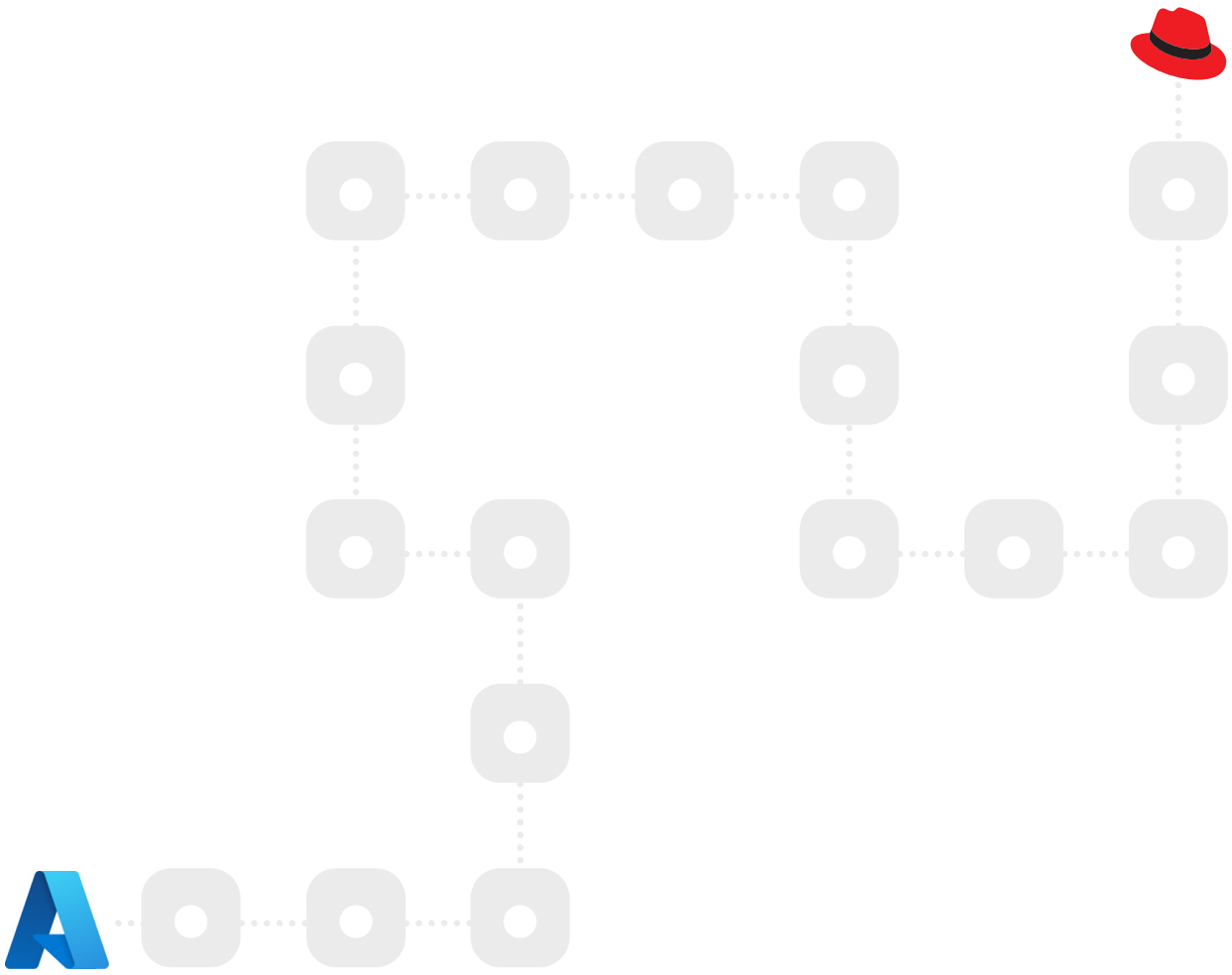
Chapitre 8 *Exploration de la plateforme d'applications :* fournit un aperçu guidé de certaines fonctionnalités clés de la plateforme d'applications : registre de conteneurs, pipelines, serverless, etc.

Chapitre 9 *Intégration à d'autres services :* couvre l'intégration des plateformes via Azure Service Operator, Azure DevOps et d'autres services similaires.

Chapitre 10 *Intégration des charges de travail et des équipes :* complète ce manuel en fournissant quelques bonnes pratiques et recommandations sur la manière d'intégrer les équipes chargées des applications et d'obtenir un certain succès dans l'adoption du service au sein de votre organisation.

Chapitre 11 *Conclusion :* contient la conclusion.

Chapitre 12 *Glossaire :* contient le glossaire.



Chapitre 2

Présentation de Red Hat OpenShift

Ce chapitre décrit brièvement Red Hat OpenShift, son utilisation et les avantages du service que les clients de Red Hat mettent généralement en avant. Cette introduction est utile si vous découvrez OpenShift pour la première fois. Elle permet également d'actualiser vos connaissances si vous cherchez à en savoir un peu plus sur la plateforme.

Présentation de la solution Red Hat OpenShift

La solution Red Hat OpenShift est une plateforme d'applications prête à l'emploi pour les entreprises, qui automatise l'exploitation de toute la pile pour la gestion des déploiements multicloud et de clouds hybrides. Elle est optimisée pour augmenter la productivité des développeurs et favoriser l'innovation. Avec une exploitation automatisée et une gestion rationalisée du cycle de vie, Red Hat OpenShift permet aux équipes de développement de créer et de déployer des applications. Les équipes d'exploitation, quant à elles, peuvent provisionner, gérer et mettre à l'échelle des plateformes Kubernetes.

Les équipes de développement ont accès aux images et aux solutions validées de nombreux partenaires. En outre, une analyse de la sécurité et des signatures cryptographiques sont effectuées tout au long du processus de distribution. Elles peuvent accéder à des images à la demande et obtenir un accès natif à une large gamme de services cloud tiers, le tout au moyen d'une plateforme unique.

Les équipes d'exploitation ont une visibilité sur les déploiements, indépendamment de leur emplacement, et sur les différentes équipes, grâce à la journalisation et à la surveillance intégrées. Les opérateurs Kubernetes Red Hat intègrent la logique d'application unique qui permet au service d'être fonctionnel. Ce dernier est non seulement configuré, mais aussi optimisé pour les performances. Il est également mis à jour et reçoit des correctifs à partir du système d'exploitation à l'aide d'une seule action. En résumé, Red Hat OpenShift est un guichet unique qui permet aux équipes informatiques et de développement d'augmenter leur productivité.

Services cloud OpenShift – Économies et avantages commerciaux

Des milliers de clients font confiance à Red Hat OpenShift pour transformer la façon dont ils distribuent des applications, améliorer les relations avec leurs clients et développer un avantage concurrentiel. Leur objectif : devenir des leaders de leur secteur. L'étude d'IDC, décrite dans La valeur ajoutée de Red Hat OpenShift pour les entreprises, mars 2021, démontre la forte valeur ajoutée dont les organisations interrogées ont bénéficié avec la plateforme Red Hat OpenShift grâce à la fourniture d'applications et de fonctionnalités de meilleure qualité et plus rapides à leurs entreprises, ainsi qu'à l'optimisation des coûts de développement et liés à l'informatique, et des besoins en temps du personnel.

Principaux indicateurs de mesure :

- Retour sur investissement de 636 % sur 5 ans
- Délai d'amortissement de 10 mois
- 54 % de réduction des coûts d'exploitation sur cinq ans
- 3 fois plus de nouvelles fonctionnalités par an
- 20 % de gain de productivité pour les développeurs d'applications
- Réduction de 71 % des temps d'arrêt non planifiés
- 21 % de gain de productivité pour les équipes chargées de l'infrastructure informatique

Source : livre blanc d'IDC, commissionné par Red Hat, *« La valeur ajoutée de Red Hat OpenShift pour les entreprises »*, document n° US47539121, mars 2021

Les services cloud Red Hat OpenShift, qui comprennent Azure Red Hat OpenShift, s'appuient sur Red Hat OpenShift et offrent des avantages supplémentaires pour l'entreprise. L'étude The Total Economic Impact™ of Red Hat OpenShift Cloud Services – Cost Savings and Business Benefits, réalisée par Forrester, met en évidence plusieurs avantages financiers clés.

Principaux avantages financiers des services cloud OpenShift :

- Retour sur investissement (ROI) de 468 %
- Valeur actualisée nette de 4,08 millions de dollars
- Délai d'amortissement de 6 mois

Outre ces avantages financiers, voici les principales conclusions des avantages quantifiés du rapport Forrester :

- **Amélioration de la vitesse de développement** : l'utilisation des services cloud Red Hat OpenShift permet aux entreprises de raccourcir leur cycle de développement jusqu'à 70 %.
- **Gain de 20 % sur le temps que les développeurs allouent aux opérations de maintenance de l'infrastructure** : selon les personnes interrogées, grâce aux services cloud Red Hat OpenShift, les développeurs n'étaient plus contraints d'assurer la maintenance de l'infrastructure de développement des applications, ce qui leur permettait de se concentrer pleinement sur l'élaboration du produit ou de la solution. Ce gain de temps représente une économie de plus de 2,3 millions de dollars sur 3 ans.
- **Amélioration de l'efficacité opérationnelle de 50 %** : les services cloud Red Hat OpenShift étant un service géré, les personnes interrogées ont indiqué que l'utilisation de la solution leur permettait de réaffecter 50 % des employés DevOps précédemment en charge de la gestion de l'infrastructure à d'autres tâches plus productives. Cette efficacité opérationnelle accrue se chiffre à plus de 1,3 million de dollars en 3 ans.*

Avantages non quantifiés que le rapport met également en avant :

- **Satisfaction et rétention des développeurs** : selon les personnes interrogées, les services cloud Red Hat OpenShift ont permis aux développeurs de décomposer les mises à jour en éléments plus petits afin de réduire la pression induite par l'exécution de tests approfondis dans un délai très court et la nécessité de répondre aux exercices d'alerte après la mise en production.
- **Sécurité et réduction des risques** : les personnes interrogées ont partagé la façon dont les services cloud Red Hat OpenShift ont automatisé certaines fonctionnalités et mises à jour de sécurité, supprimant ainsi la nécessité d'une maintenance manuelle tout en garantissant la sécurité de leur environnement.
- **Fiabilité** : selon les personnes interrogées, l'utilisation des services cloud Red Hat OpenShift a renforcé la fiabilité de leur plateforme d'applications à long terme en diminuant les pannes ou les défaillances du système, y compris avec un environnement évolutif.
- **Portabilité et continuité de l'activité** : selon les personnes interrogées, les services cloud Red Hat OpenShift garantissaient la continuité des activités et participaient à leur stratégie de récupération après sinistre grâce à leur portabilité, leur évolutivité et leur flexibilité.

Source : Forrester, *The Total Economic Impact of Red Hat OpenShift Cloud Services*, décembre 2021

***Pour en savoir plus :** Une entreprise plus agile grâce aux services cloud

« Red Hat OpenShift ou Kubernetes nu ? » – Le coût de la création de votre propre plateforme d'applications Kubernetes

Red Hat OpenShift est souvent qualifié de « Kubernetes d'entreprise », mais la signification de cette expression peut ne pas être claire de prime abord. Les clients demandent souvent : « OpenShift ou Kubernetes nu ? ». Il faut toutefois comprendre qu'**OpenShift utilise déjà Kubernetes**. Dans l'architecture globale d'OpenShift, Kubernetes fournit à la fois la base sur laquelle la plateforme OpenShift est construite et une grande partie des outils permettant d'exécuter OpenShift.

Kubernetes est un projet open source crucial, l'un des projets clés de la Cloud Native Computing Foundation et une technologie essentielle pour l'exploitation des conteneurs.

Toutefois, les utilisateurs potentiels d'OpenShift pourraient sérieusement se demander : « **Puis-je exécuter mes applications avec Kubernetes uniquement ?** » De nombreuses organisations commencent par déployer Kubernetes et constatent qu'elles peuvent faire fonctionner un conteneur, voire une application d'entreprise, en quelques jours seulement. Cependant, lorsque les opérations du jour 2 commencent, que les exigences en matière de sécurité apparaissent et que davantage d'applications sont déployées, certaines organisations tombent dans le piège de la construction de leur propre **plateforme en tant que service (PaaS)** avec la technologie Kubernetes. Elles ajoutent un contrôleur open-source entrant, écrivent quelques scripts pour se connecter à leurs **pipelines d'intégration continue/déploiement continu (CI/CD)**, puis essaient de déployer une application plus complexe... et c'est là que les problèmes commencent. Si le déploiement de Kubernetes était la partie émergée de l'iceberg, la complexité de la gestion du jour 2 en serait la partie cachée, imposante et immergée.

S'atteler à la résolution de ces défis et problèmes est possible, mais requiert généralement une équipe d'exploitation de plusieurs personnes, et plusieurs semaines et mois d'efforts pour créer un « PaaS personnalisé conçu sur Kubernetes » et en assurer la maintenance. Il en résulte un manque d'efficacité au sein de l'organisation, une prise en charge complexe du point de vue de la sécurité et de la certification, et la nécessité de tout développer à partir de zéro lors de l'intégration des équipes de développeurs. Si une organisation devait dresser la liste de toutes les tâches liées à la mise en place et à la maintenance d'une telle plateforme Kubernetes personnalisée (c'est-à-dire Kubernetes) en intégrant tous les composants supplémentaires nécessaires à l'exécution réussie des conteneurs, voici ce qu'elle contiendrait :

- **Gestion des clusters** : comprend l'installation de systèmes d'exploitation, l'application de correctifs au système d'exploitation, l'installation de Kubernetes, la configuration de la mise en réseau de l'interface CNI, l'intégration de l'authentification, la configuration des entrées et des sorties, la configuration du stockage persistant, le renforcement des nœuds, l'application de correctifs de sécurité et la configuration du cloud/multicloud sous-jacent.
- **Services d'applications** : comprennent l'agrégation des journaux, les contrôles d'intégrité, la surveillance des performances, les correctifs de sécurité, le registre de conteneurs et la configuration du processus de mise à disposition des applications.
- **Intégration des développeurs** : comprend l'intégration CI/CD, l'intégration des outils de développement/IDE, l'intégration du framework, la compatibilité du middleware, la fourniture de tableaux de bord sur les performances des applications et les contrôles d'accès basés sur les rôles (RBAC).

Bien qu'il soit possible d'ajouter de nombreuses autres actions et technologies à cette liste (la plupart de ces activités sont essentielles pour que toute organisation utilise sérieusement les conteneurs), la complexité, le temps et les efforts nécessaires à leur mise en place sont insignifiants par rapport à la maintenance continue de ces éléments individuels. Chaque intégration doit faire l'objet de tests approfondis, et chaque composant et activité aura un cycle de publication, une politique de sécurité et des correctifs différents.

Avantages de Red Hat OpenShift par rapport à Kubernetes nu ?

Lorsqu'une organisation en vient à exécuter des conteneurs en production, conçus sur un Kubernetes nu, les composants mentionnés dans la section précédente sont normalement installés et intégrés ensemble pour créer une sorte de plateforme d'applications ayant sa propre structure.

Azure Red Hat OpenShift intègre tous les composants mentionnés dans la section précédente dans une seule plateforme. Les équipes informatiques tirent ainsi parti d'une plus grande facilité d'exploitation, tandis que les équipes des applications ont à leur disposition tout ce dont elles ont besoin pour exécuter leurs tâches. Nous aborderons tous ces sujets en détail plus loin dans ce manuel. En attendant, examinons certaines différences majeures qui existent entre les deux :

- **Facilité de déploiement :** le déploiement d'une application dans Kubernetes peut s'avérer chronophage. Vous devez extraire votre code GitHub sur une machine, mettre en route un conteneur, l'héberger dans un registre comme Docker Hub et enfin comprendre votre pipeline CI/CD, ce qui peut être très compliqué. En revanche, OpenShift automatise les plus grosses tâches et le travail de back-end. Il ne vous reste plus qu'à créer un projet et à télécharger votre code.
- **Sécurité :** aujourd'hui, nous constatons que la plupart des projets Kubernetes sont réalisés par des équipes de plusieurs développeurs et opérateurs. Même si Kubernetes prend désormais en charge des éléments tels que les contrôles RBAC et la gestion des identités et des accès (IAM), une installation et une configuration manuelles sont encore nécessaires et s'avèrent chronophages. Grâce à des années d'expérience, un important travail d'identification des meilleures pratiques relatives à la sécurité a été mené avec Red Hat et OpenShift. Les clients peuvent dorénavant en tirer parti, car elles sont prêtes à l'emploi. Vous vous contentez d'ajouter de nouveaux utilisateurs et OpenShift se charge de gérer certains éléments comme l'espacement des noms et la création de différentes politiques de sécurité.
- **Flexibilité :** avec Azure Red Hat OpenShift, vous pouvez tirer parti des meilleures pratiques bien connues de déploiement, de gestion et de mise à jour. Toutes les grosses tâches dans le back-end sont effectuées pour vous de manière très fluide. Vous pouvez ainsi vous consacrer plus rapidement au développement de vos applications. Outre que certaines équipes apprécient qu'on leur explique comment travailler et tirent parti d'une approche rationalisée, la plateforme Kubernetes vous permet de personnaliser manuellement votre pipeline de développement CI/CD, ce qui laisse davantage de place pour la flexibilité et la créativité lors du développement de vos processus.

- **Exploitation au quotidien :** les clusters sont constitués d'un groupe de plusieurs machines virtuelles et inévitablement, vos équipes d'exploitation devront créer des machines virtuelles qui doivent être ajoutées à un cluster. Le processus de configuration au moyen de Kubernetes peut s'avérer chronophage et complexe. Il implique le développement de scripts pour configurer des éléments tels que l'auto-inscription ou l'automatisation du cloud. Avec Azure Red Hat OpenShift, les opérations de provisionnement de cluster, de mise à l'échelle et de mise à niveau sont automatisées et gérées par la plateforme.
- **Gestion** : bien que vous puissiez tirer parti des outils et tableaux de bord standard de Kubernetes qui sont fournis avec n'importe quelle distribution, la plupart des développeurs requièrent une plateforme plus complète et robuste. Azure Red Hat OpenShift offre une excellente console web qui repose sur les API Kubernetes, ainsi que des fonctionnalités permettant aux équipes d'exploitation de gérer leurs charges de travail.

Le *chapitre 8, Exploration de la plateforme d'applications*, propose une description guidée de plusieurs fonctionnalités à valeur ajoutée incontournables d'Azure Red Hat OpenShift.

Synthèse

De nombreux clients communs de Red Hat et de Microsoft choisissent Azure Red Hat OpenShift comme plateforme d'applications préférée pour adopter des applications conteneurisées.

Le chapitre suivant présente Red Hat OpenShift en tant que service cloud. Nous découvrirons son architecture, son intégration et sa gestion.

Chapitre 3

Azure Red Hat OpenShift

Avec **Azure Red Hat OpenShift**, vous pouvez déployer des clusters Red Hat OpenShift entièrement gérés sans devoir créer ni gérer son infrastructure d'exécution.

Azure Red Hat OpenShift est une plateforme d'applications à la demande, cloud-native, conçue, gérée et prise en charge conjointement par Red Hat et Microsoft. Une équipe spécialisée en **ingénierie de la fiabilité des sites** (**SRE**) automatise, met à l'échelle et sécurise les clusters OpenShift et travaille côte à côte pour offrir une expérience de support intégrée. Avec Azure Red Hat OpenShift, vous n'avez aucune machine virtuelle à exploiter et aucun correctif à appliquer. Red Hat et Microsoft se chargent de l'application de correctifs aux nœuds de plan de contrôle et de calcul, ainsi que de leur mise à jour et de leur surveillance à votre place. Vos clusters Azure Red Hat OpenShift sont déployés dans le cadre de votre abonnement Azure et sont inclus dans votre facture Azure.

Fournissez des applications plus rapidement avec Azure Red Hat OpenShift :

- Les développeurs ont les moyens d'innover grâce aux pipelines CI/CD intégrés. Ils peuvent ensuite facilement connecter vos applications à des centaines de services Azure tels que MySQL, PostgreSQL, Redis et Azure Cosmos DB.
- L'automatisation du provisionnement, de la configuration et des opérations simplifie le fonctionnement tout en supprimant les obstacles à la productivité.
- Selon vos conditions, vous pouvez démarrer un cluster hautement disponible avec trois nœuds de calcul en quelques minutes, puis le mettre à l'échelle en fonction des variations de la demande d'applications en choisissant parmi des nœuds de calcul standard, à haute mémoire ou à haute puissance de calcul.
- Vous bénéficiez de fonctionnalités d'exploitation, de sécurité et de conformité adaptées pour l'entreprise, complétées par un système d'assistance intégré.

Nous étudierons ensuite les détails techniques de la conception et du fonctionnement de Red Hat OpenShift.

Architecture

Azure Red Hat OpenShift utilise les services d'infrastructure Azure (machines virtuelles, groupes de sécurité réseau, comptes de stockage et autres services Azure) comme support de base pour installer Red Hat OpenShift. L'architecture Azure est également entièrement déployée dans votre souscription Azure, ce qui facilite l'intégration à d'autres services déjà opérationnels sur votre compte.

L'interface OpenShift elle-même est conçue sur Red Hat Enterprise Linux CoreOS, qui héberge l'architecture basée sur les microservices, composée d'unités plus petites et découplées qui travaillent ensemble. Le service `kubelet` est exécuté sur chaque machine virtuelle, constituant ainsi la base du cluster Kubernetes. La base de données sous-jacente à cette architecture, qui s'exécute sur le plan de contrôle, est **etcd**, un magasin de clés-valeurs fiable et clusterisé.

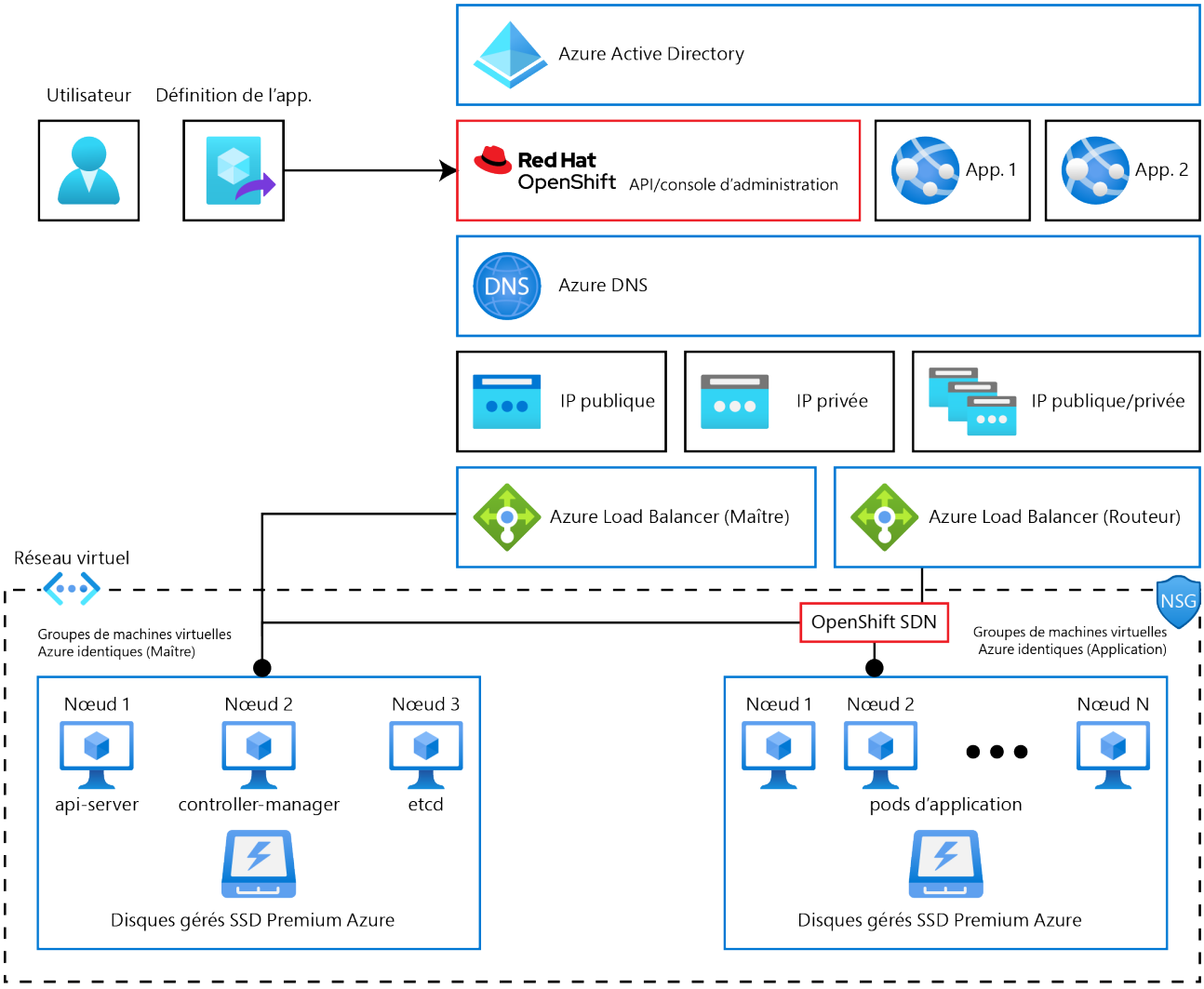


Figure 3.1 : Architecture d'Azure Red Hat OpenShift

Les deux sections suivantes, *Calcul – nœuds de contrôle et de calcul* et *Mise en réseau*, expliquent plus en détail le contenu de ce diagramme.

Calcul – nœuds de contrôle et de calcul

L'architecture de Red Hat OpenShift fonctionne sur des machines virtuelles (nœuds) qui assurent l'un des trois rôles spécifiques suivants : contrôle, infrastructure ou application. Cependant, la version 4 d'Azure Red Hat OpenShift ne prenant pas en charge les nœuds d'infrastructure au moment de la rédaction de ce document, vous n'y trouverez pas d'informations sur les nœuds de contrôle et de calcul.

Les nœuds de **contrôle** (historiquement appelés nœuds **maîtres**) sont des machines virtuelles qui contiennent des composants essentiels pour le cluster Kubernetes. Il s'agit notamment du serveur d'API, du serveur du gestionnaire de contrôleurs, du planificateur et du magasin etcd. Ils gèrent le cluster Kubernetes et programment les pods à exécuter sur les nœuds de calcul.

- **Le serveur d'API Kubernetes** valide et configure les données pour les pods, les services et les contrôleurs de réplication. Il fournit également un point focal pour l'état partagé du cluster.
- **Le gestionnaire de contrôleurs Kubernetes** surveille le magasin etcd pour détecter les modifications apportées aux objets, tels que les objets de réplication, d'espace de noms et de contrôleur de compte de service, puis il utilise l'API pour appliquer l'état spécifié. Plusieurs de ces processus créent un cluster avec un leader actif à la fois.
- **Le planificateur Kubernetes** surveille les pods nouvellement créés sans nœud attribué et sélectionne le meilleur nœud pour héberger le pod.
- **Le magasin etcd** stocke l'état persistant du maître, tandis que les autres composants surveillent les modifications apportées dans etcd pour adapter leur état en conséquence.

Vos applications sont exécutées sur les nœuds d'**application**.

Chaque nœud d'un cluster Kubernetes exécute un service appelé `kubelet`, qui gère l'**interface d'exécution de conteneur** (**cri-o**), un proxy de service et d'autres services essentiels exécutés sur chaque nœud. La technologie de mise en réseau logicielle d'OpenShift connecte tous les nœuds. Dans Azure Red Hat OpenShift, il s'agit d'un **réseau virtuel ouvert (OVN)**, qui fonctionne sur un réseau virtuel Azure.

Azure Red Hat OpenShift crée des nœuds qui fonctionnent sur des machines virtuelles Azure connectées à des disques de stockage Azure. Vous remarquerez que les disques sont plutôt volumineux (1 To). En effet, dans Azure, la garantie IOPS relative aux performances de stockage dépend de la taille du disque. Une taille de 1 To offre une bande passante suffisante pour le disque sous-jacent utilisé par la base de données etcd.

Mise en réseau

Azure Red Hat OpenShift nécessite un seul réseau virtuel Azure avec deux sous-réseaux configurés : un pour les nœuds du plan de contrôle et un pour les nœuds de calcul. La taille minimale des deux réseaux est de /27 (32 adresses). Cependant, veillez à ne pas définir une taille trop petite si vous avez l'intention de faire évoluer le cluster ultérieurement.

Les deux équilibreurs de charge Azure (appelés **Maître** et **Routeur** dans le diagramme) dirigent le trafic comme suit :

- Équilibreur de charge maître/plan de contrôle : envoi du trafic entrant pour les utilisateurs de l'API OpenShift. Fournit également une connectivité sortante pour les nœuds du plan de contrôle.
- Routeur/équilibreur de charge d'application : envoi du trafic d'entrée aux applications exécutées dans OpenShift, via un « routeur » OpenShift ou un contrôleur d'entrée. Fournit également une connectivité sortante pour les nœuds de calcul.

Un excellent article sur la configuration, les exigences et les limites du réseau est disponible dans la documentation sur le réseau.

Intégration à d'autres services Azure

Vous pouvez déployer Azure Red Hat OpenShift en tant que service natif sur Azure et l'intégrer à de nombreux services que vous avez l'habitude d'utiliser sur Azure. Voici un aperçu global de quelques-uns des services d'infrastructure Azure pour lesquels des intégrations communes existent.

La *figure 3.2* présente de nombreux points d'intégration communs des services Azure pour OpenShift sur Azure :

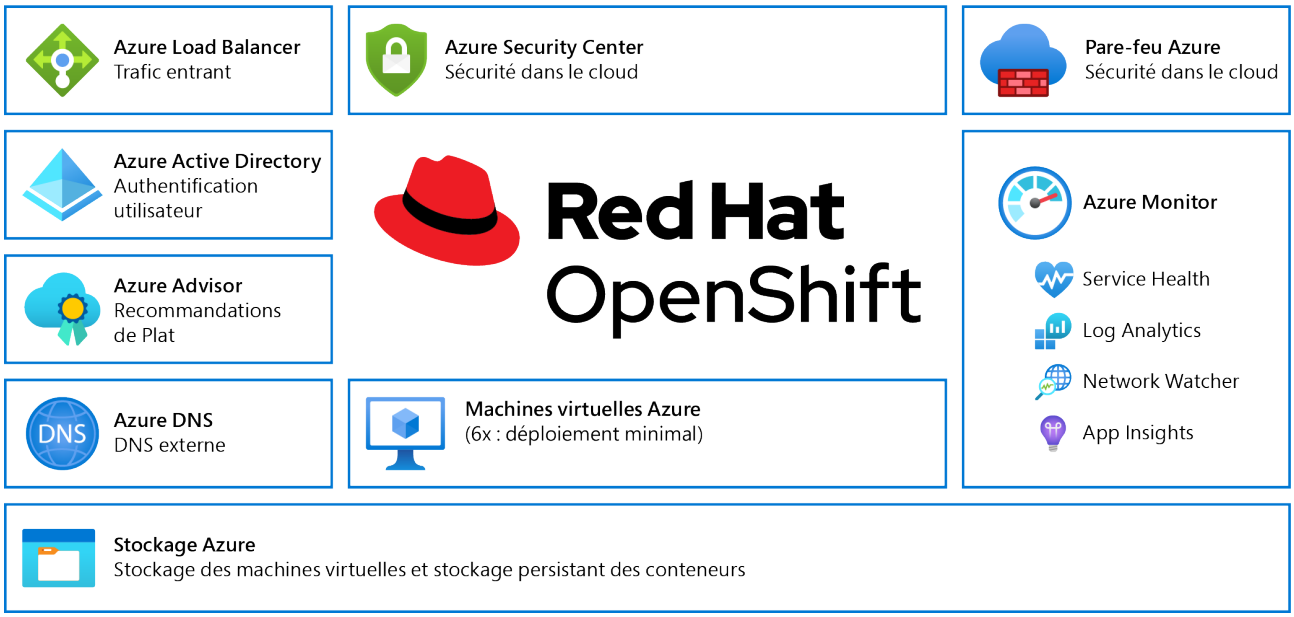


Figure 3.2 : Points d'intégration communs des services Azure

En outre, les applications exécutées sur OpenShift peuvent s'intégrer très étroitement aux services Azure via Azure Service Operator. Ce point est abordé plus en détail dans le *chapitre 9, Intégration à d'autres services*.

Gestion

Pour comprendre les éléments gérés dans le cadre du service Azure Red Hat OpenShift, considérez que tous les éléments, du datacenter aux opérateurs de clusters, sont « gérés ». Les opérateurs de cluster sont des services qui s'exécutent sur les nœuds du plan de contrôle et prennent en charge la surveillance, les mises à jour et l'intégrité du cluster. Cela signifie que Microsoft et Red Hat assureront conjointement la surveillance, la maintenance et la gestion de ces composants afin de maintenir le cluster en ligne. Tous les éléments au-dessus des opérateurs de cluster restent à la charge du client.

En tant qu'utilisateur d'Azure Red Hat OpenShift, vous disposez d'un accès complet au **cluster en tant qu'administrateur**, ce qui signifie que vous partagez également la responsabilité de ne pas nuire au fonctionnement de certaines parties de votre cluster. Vous devez comprendre la politique de support pour savoir ce que vous pouvez et ne pouvez pas faire avec le cluster.

La matrice de responsabilité Azure Red Hat OpenShift décrit de manière détaillée le rôle de Microsoft et de Red Hat dans le cadre du service cloud.

Authentification et autorisation

Azure Active Directory est un moyen courant de fournir une authentification aux clusters Azure Red Hat OpenShift. Il n'est toutefois pas obligatoire. Vous pouvez également utiliser d'autres mécanismes d'authentification, tels que la connexion avec GitHub, ou un simple « fichier de mots de passe ».

Lorsqu'Azure Active Directory est utilisé, les API d'Azure Red Hat OpenShift et Kubernetes transmettront les demandes d'authentification. Les utilisateurs présenteront leurs informations d'identification et recevront une autorisation en fonction de leurs rôles.

La couche d'authentification identifie l'utilisateur associé aux requêtes adressées à l'API Azure Red Hat OpenShift. La couche d'autorisation utilise ensuite les informations sur l'utilisateur à l'origine de la demande pour déterminer si celle-ci doit être autorisée.

La section *Authentification – Azure Active Directory* décrit les instructions de configuration d'Azure Active Directory. Ce processus est très similaire d'un point de vue fonctionnel si vous utilisez un autre fournisseur d'authentification qu'Azure Active Directory.

L'autorisation est gérée dans le moteur de politique Azure Red Hat OpenShift, qui définit des actions telles que « créer un pod » ou « répertorier les services » et les regroupe en rôles dans un document de politique. Les rôles sont liés aux utilisateurs ou aux groupes par l'identifiant de l'utilisateur ou du groupe. Lorsqu'un utilisateur ou un compte de service entreprend une action, le moteur de politique recherche un ou plusieurs des rôles attribués à l'utilisateur (par exemple, l'administrateur client ou l'administrateur du projet en cours) avant de l'autoriser à poursuivre.

Les relations entre les rôles de cluster, les rôles locaux, les liaisons de rôle de cluster, les liaisons de rôle local, les comptes d'utilisateurs, de groupes et de service sont illustrées sur la *figure 3.3* :

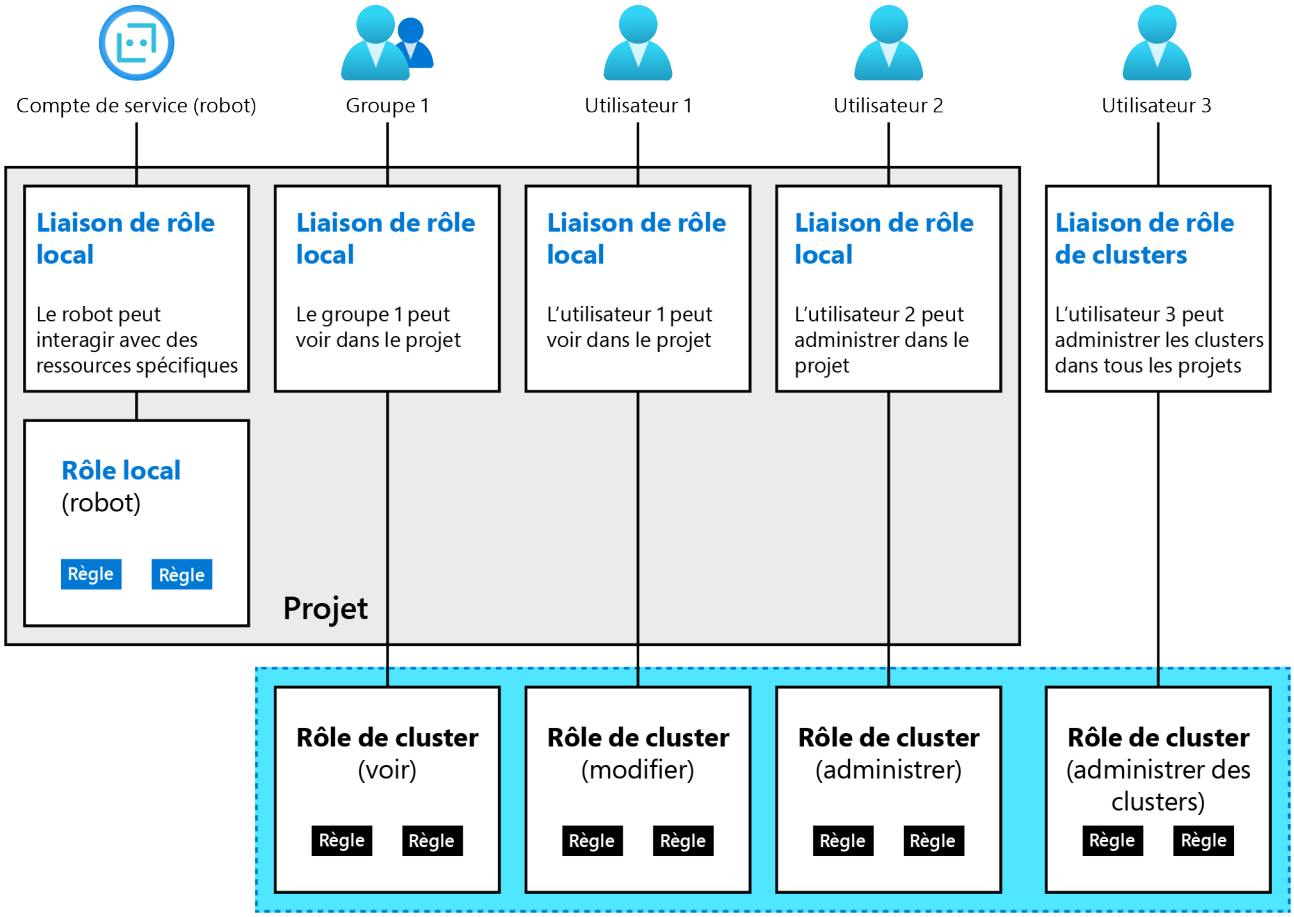


Figure 3.3 : Relations entre les rôles

La documentation de Red Hat OpenShift contient une liste complète des fournisseurs d'authentification.

Assistance

La gestion de l'assistance d'Azure Red Hat OpenShift est unique. Les équipes d'assistance de Microsoft et de Red Hat travaillent ensemble, aux côtés de l'équipe mondiale d'ingénierie de la fiabilité des sites (SRE), pour faciliter le fonctionnement du service.

Les clients demandent une assistance dans le portail Azure. Les demandes sont alors triées et traitées par les ingénieurs Microsoft et Red Hat afin d'y répondre rapidement, qu'elles soient au niveau de la plateforme Azure ou d'OpenShift.

Voici un aperçu du processus d'assistance intégré :

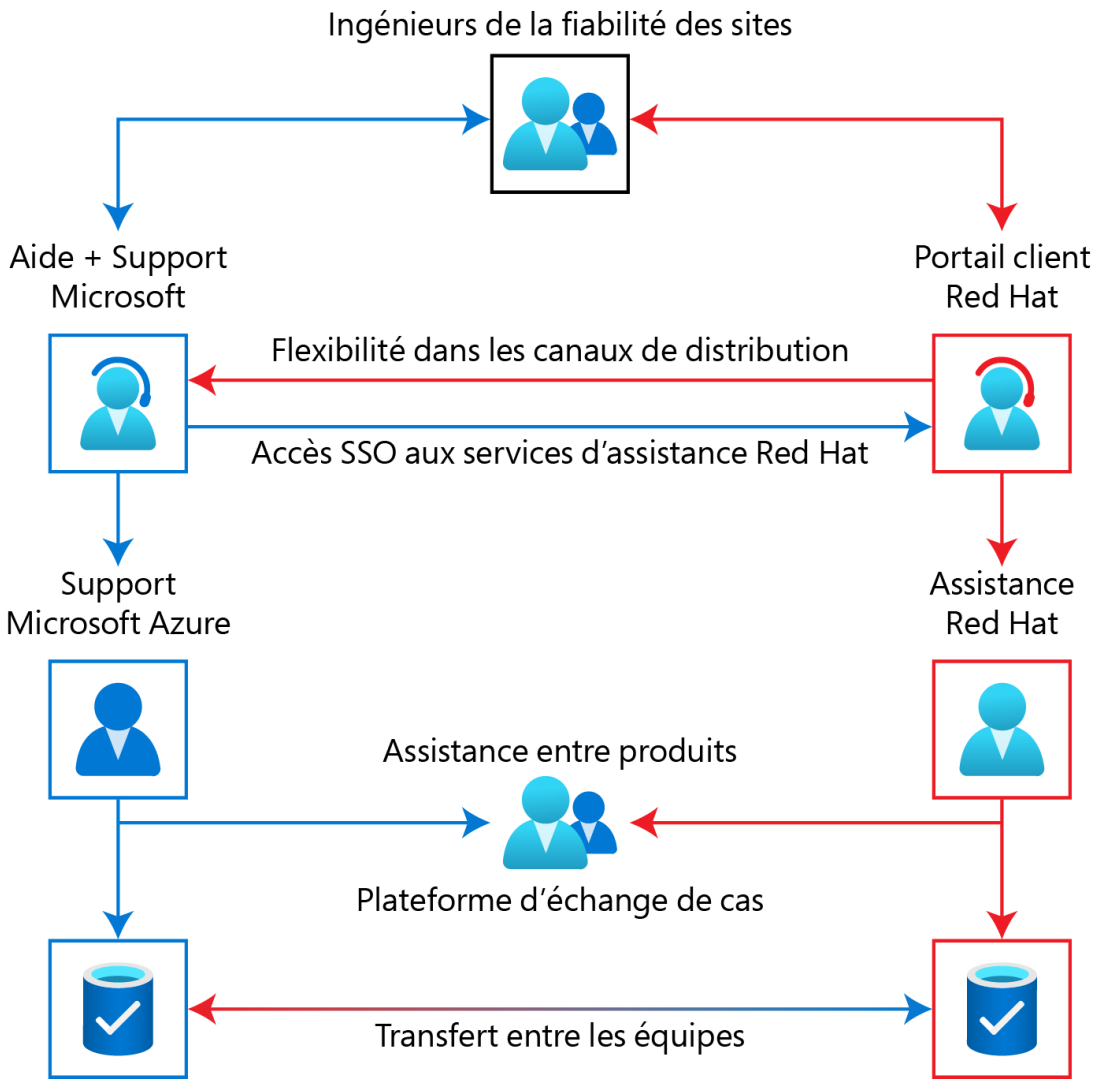


Figure 3.4 : Processus d'assistance intégré

La figure 3.4 montre que les clients peuvent créer un ticket d'assistance sur le portail d'assistance Microsoft ou sur le portail d'assistance Red Hat. Notez que pour accéder au portail d'assistance Red Hat, le cluster doit être enregistré dans OpenShift Cluster Manager.

Les ingénieurs d'assistance de Microsoft peuvent collaborer de manière transparente avec Red Hat à n'importe quel stade, avec le consentement d'un client via la plateforme d'échange de cas. Il en va de même pour les ingénieurs d'assistance de Red Hat lorsqu'ils demandent une collaboration avec Microsoft. Les ingénieurs d'assistance des deux entreprises peuvent contacter l'équipe SRE. Cette dernière peut prendre des mesures correctives pour réparer les clusters, si nécessaire.

Prix et souscriptions

Par rapport à une installation « Do It Yourself » (DIY), Azure Red Hat OpenShift offre notamment comme avantage majeur que le calcul, le réseau, le stockage et l'infrastructure Azure, ainsi que les souscriptions OpenShift, sont tous facturés via votre souscription Azure et non séparément. Pour vous faire une idée indicative des coûts de la solution, ajoutez simplement Azure Red Hat OpenShift au générateur de devis dans le [calculateur de prix Azure](#).

Voici un manuel explicatif sur la page de tarification :

1. Tapez *openshift* dans le champ de recherche et ajoutez-le à un nouveau devis.

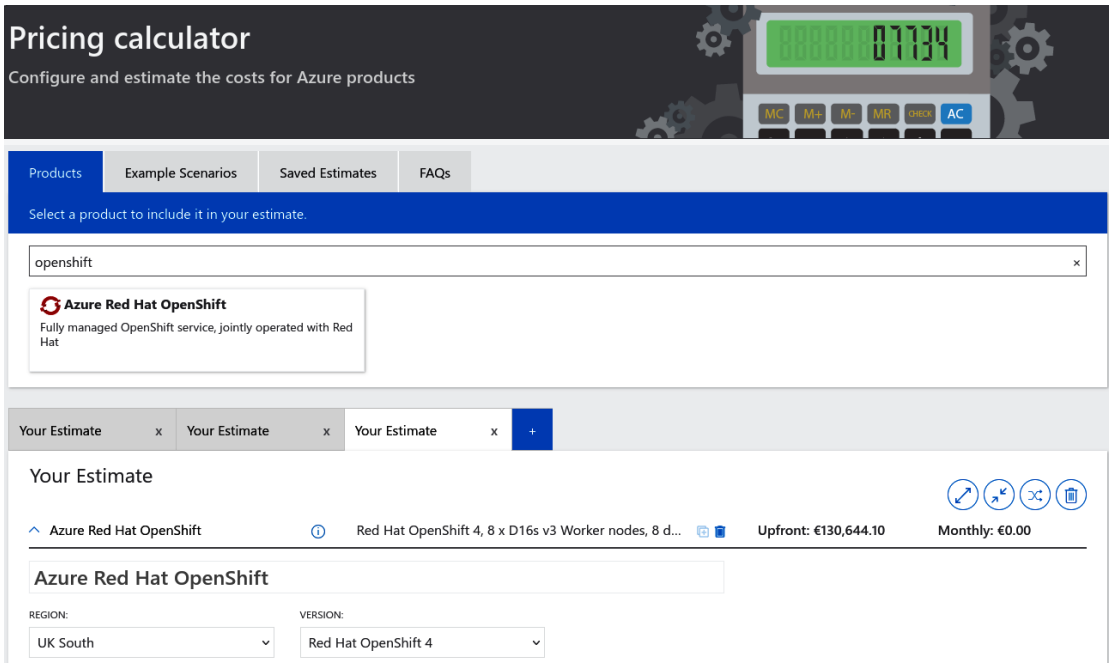


Figure 3.5 : Interface du calculateur de prix

2. Au bas de la page, vous trouverez une liste déroulante permettant de remplacer l'unité de prix par votre devise locale. Cet exemple utilise la devise GBP (£).
3. Définissez votre région Azure pour le déploiement d'Azure Red Hat OpenShift. Le prix variera légèrement d'une région à l'autre pour tenir compte des différents coûts de calcul entre les régions Azure.

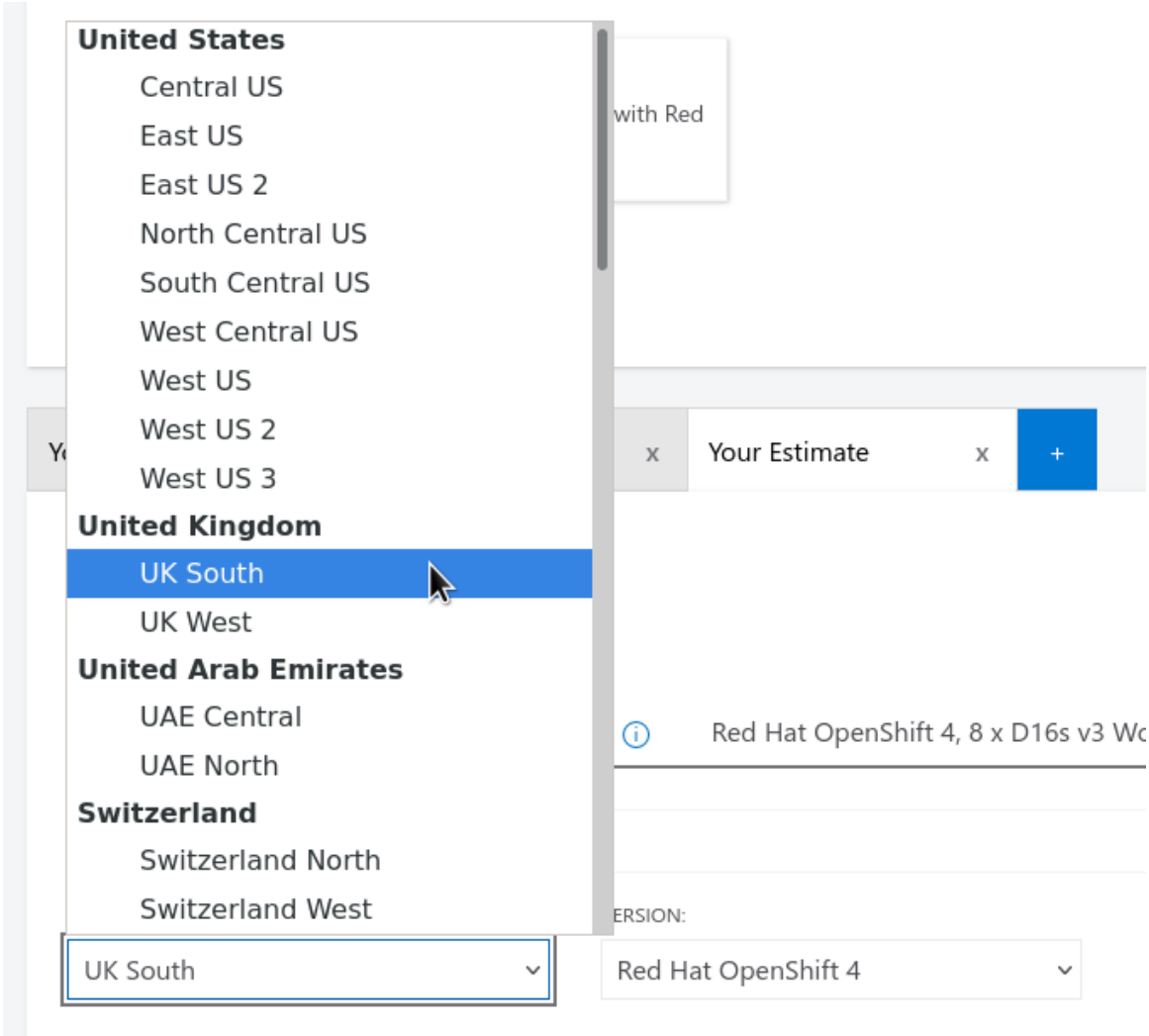


Figure 3.6 : Sélection de la région Azure

4. Vos applications réelles seront exécutées sur les **nœuds de calcul**. **Savings Options** (Options économiques) contient deux sections : **License** (Licence) fait référence au coût des souscriptions OpenShift ; **Virtual Machine** (Machine virtuelle) fait référence au coût des services de calcul requis pour exécuter le cluster. Le nombre maximal de nœuds de travail pris en charge dans un cluster est de 100.

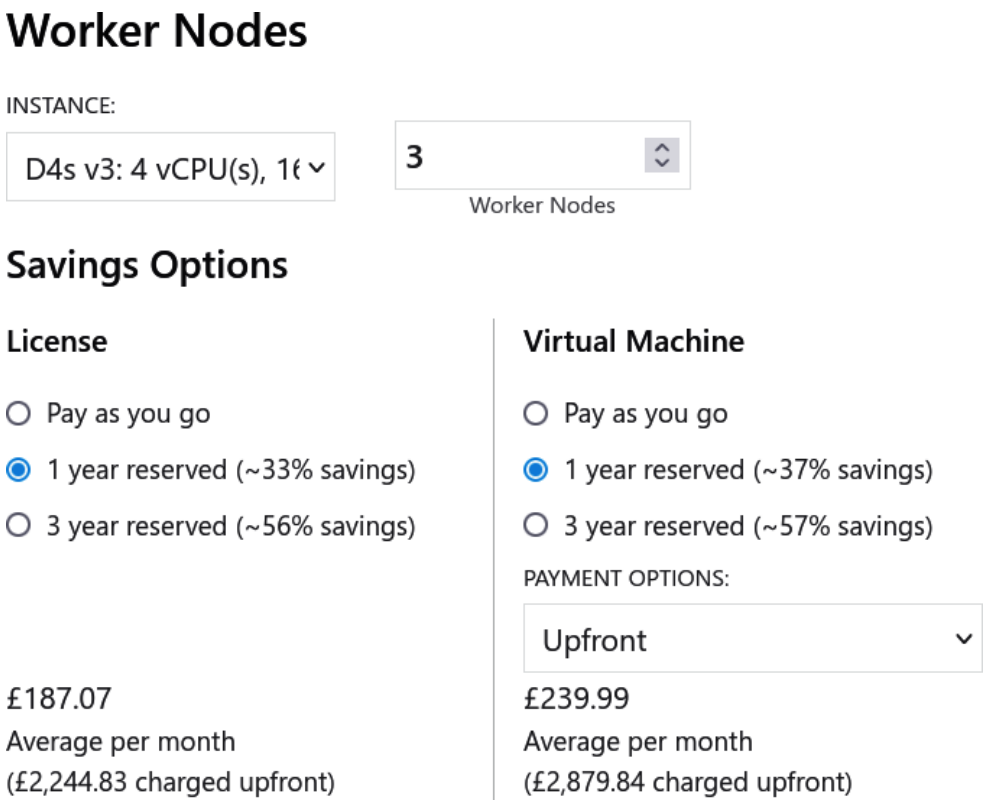


Figure 3.7 : Détails des nœuds de calcul

5. **Managed OS Disks** (Disques de système d'exploitation gérés) fait référence à la taille des disques utilisés par Red Hat CoreOS pour les partitions du système d'exploitation. Cette option ne fait pas référence au stockage utilisé par les volumes persistants des applications, qui sont provisionnés séparément.

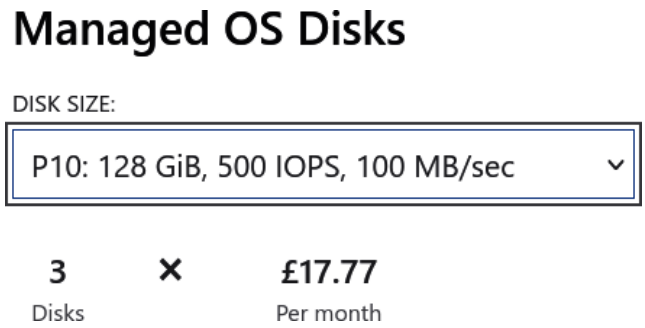


Figure 3.8 : Disques de système d'exploitation gérés

6. Une section similaire existe pour les **nœuds maîtres**, qui sont généralement appelés nœuds de contrôle dans le langage moderne. Notez que les nœuds de contrôle sont associés à un disque nettement plus grand que les nœuds de calcul. Cela s'explique par le fait qu'ils requièrent des IOPS et un débit plus élevés, impliquant des disques de plus grande taille sur Azure. Le nombre exact de nœuds du plan de contrôle doit toujours être de trois, pour la stabilité du cluster.

Notez que le calculateur est conçu pour afficher des prix indicatifs, les prix réels fluctueront en fonction de l'utilisation.

Azure : instances de machines virtuelles réservées

Une **instance réservée** fait référence à un engagement d'utilisation de cette ressource pendant une certaine durée : 1 ou 3 ans. Il existe des options d'instances réservées pour les machines virtuelles Azure Red Hat OpenShift.

Les organisations choisissent généralement des instances réservées pour obtenir une réduction substantielle sur l'IaaS, lorsqu'elles savent qu'un cluster va fonctionner pendant une longue période. Par exemple, l'exécution des environnements de production s'appuie souvent sur des instances réservées. Notez que les instances réservées ne modifient pas le niveau de service ni l'architecture du cluster.

[En savoir plus sur les instances réservées Azure]

Synthèse

Ce chapitre a présenté les spécificités du service cloud géré Azure Red Hat OpenShift. Il a donné un aperçu global de l'architecture et abordé brièvement l'intégration à d'autres services Azure (que le *chapitre 9, Intégration à d'autres services* traitera plus en détail), ainsi que les considérations relatives à la gestion, à l'authentification, à l'assistance et à la tarification.

Le chapitre suivant se concentrera sur les questions et les décisions auxquelles une organisation devra répondre lors de la phase de pré-provisionnement du déploiement d'Azure Red Hat OpenShift.

Chapitre 4

Pré-provisionnement – Questions sur l'architecture d'entreprise

De nombreuses organisations verront Azure Red Hat OpenShift sur le portail Azure. Il suffit de lire une fois la documentation pour déployer avec succès et très peu d'efforts supplémentaires un cluster avec Red Hat OpenShift. Toutefois, si vous consacrez un peu plus de temps à la planification des déploiements et vous posez quelques questions au préalable, vous pouvez gagner énormément de temps sur la suppression et le reprovisionnement de clusters ultérieurement.

Ce chapitre s'appuie sur une expérience concrète et pratique de travail avec de nombreux clients. Il couvre la plupart des questions types de pré-provisionnement à aborder au préalable. Contenu abordé dans ce chapitre :

- Nombre de clusters nécessaires, y compris la préproduction, la production et similaires
- Visibilité des réseaux publics et privés
- Connectivité hybride, telle que la connexion à des solutions sur site

Nous allons commencer par la méthode de calcul du nombre de clusters dont vous avez besoin.

Combien de clusters sont nécessaires ?

De nombreux modèles de déploiement existent pour OpenShift. Encore faut-il que l'organisation détermine le nombre de clusters dont elle a besoin. Si elle est le seul maître de cette décision, vous trouverez toutefois quelques conseils pour faciliter ce calcul dans les paragraphes suivants.

Étapes du cycle de vie : développement, essais, production

La plupart des organisations, indépendamment de leur taille, déploient leurs systèmes informatiques d'entreprise en respectant certaines étapes du cycle de vie. Cette approche est aussi parfois appelée un modèle de préproduction. Les trois étapes les plus courantes sont le développement, les tests et la production. L'existence de plusieurs étapes permet de tester les modifications apportées aux applications et les déploiements d'applications dans un environnement sûr avant que ces modifications ne se retrouvent dans un environnement de production. Le modèle de préproduction le plus courant, et recommandé, consiste à disposer d'au moins trois clusters Azure Red Hat OpenShift distincts

- **Développement :** étape au cours de laquelle les développeurs et les opérateurs sont autorisés à effectuer tous les tests qu'ils souhaitent. Ils peuvent utiliser un grand cluster de type « bac à sable », mais il est généralement plus sûr de se tourner vers de petits clusters de courte durée où les tests sont créés et détruits fréquemment.
- **Test :** étape de test et de validation des modifications à venir du cluster, telles que les correctifs ou les changements de configuration, avant leur mise en production. Dans certaines organisations, cette étape porte le nom de « préproduction », mais la préproduction peut aussi correspondre à un tout autre environnement.
- **Production :** étape d'exécution réelle des applications.

Certaines organisations utiliseront également des environnements supplémentaires, tels que des environnements de test d'intégration. Vous restez toutefois la seule personne à savoir combien d'environnements intermédiaires conviennent le mieux à votre organisation. En cas d'hésitation, observez d'autres applications d'entreprise similaires pour trouver des modèles de déploiement communs.

Lorsque vous utilisez Azure Red Hat OpenShift pour des applications non critiques, votre organisation peut n'avoir que deux clusters (en fusionnant le développement et les tests), voire un seul cluster qui inclut le développement, les tests et la production. Ce système présente l'avantage de limiter les coûts du cloud tout en réduisant le nombre total de clusters à gérer. Lorsqu'ils n'utilisent qu'un seul cluster, les administrateurs peuvent choisir d'attribuer des espaces de noms distincts aux étapes de développement, test et production. L'exploitation d'un seul cluster présente cependant des inconvénients :

- Les changements qui affectent l'ensemble du cluster (comme les correctifs logiciels) peuvent introduire des problèmes au niveau de la production qui auraient été détectés et évités s'ils avaient été exécutés dans un environnement de test.
- Si une application dans un environnement de test ou de développement se comporte de manière inattendue, par exemple en créant de nombreux conteneurs ou en utilisant tout l'espace disque disponible, des problèmes apparaîtront dans l'environnement de production.

Vous ne trouverez pas dans ce manuel le nombre précis de clusters parfaitement adaptés à votre situation. Comme mentionné précédemment, examinez les applications d'entreprise similaires dans votre organisation pour déterminer le nombre d'étapes de cycle de vie utilisées.

Continuité des activités, récupération après sinistre et basculement

Outre les environnements de test standard, de nombreuses organisations cherchent également à créer au moins un environnement de production de basculement, ce dernier étant utilisé en cas de défaillance catastrophique entraînant la mise hors service de l'ensemble du cluster ou de la région Azure. Cette action s'appelle généralement la **récupération après sinistre (DR)**. Cet environnement est habituellement déployé dans une région Azure différente de celle du cluster de production.

Le service cloud géré est assorti d'un **contrat de niveau de service (SLA)** garantissant une disponibilité de 99,95 %, un taux acceptable pour de nombreuses applications critiques pour l'entreprise. Toutefois, certaines applications peuvent nécessiter un SLA garantissant une disponibilité supérieure. Vous devez comprendre que vous ne pouvez pas obtenir une disponibilité supérieure à ce SLA avec un seul cluster. Demandez-vous si votre application a besoin d'un niveau de service supérieur à 99,95 % ou si celui-ci est suffisant pour votre application.

Si ce n'est pas le cas, vous pouvez atteindre des niveaux de service plus élevés (offrant une disponibilité de 99,999 %, par exemple) en déterminant un contrat de niveau de service composite à partir de l'exécution de plusieurs clusters en parallèle. En situant les clusters dans la même région (par exemple, `europe de l'ouest`) ou dans plusieurs régions (par exemple, `europe de l'ouest` et `europe du nord`), vous pouvez atteindre des niveaux de disponibilité encore plus élevés. Consultez la documentation Azure suivante pour savoir comment déterminer les SLA composites lors de l'exécution de plusieurs clusters.

[Documentation Azure sur les SLA composites]

Ce manuel n'aborde pas les déploiements de plusieurs clusters et multirégionaux d'Azure Red Hat OpenShift. En effet, la conception de ces architectures plus complexes implique de relever plusieurs défis associés à l'acheminement du trafic vers le cluster. Le choix des données à partager entre les clusters, ainsi que la manière de procéder, nécessitent également une réflexion approfondie.

Régions et zones de disponibilité

Le service Azure Red Hat OpenShift est conçu pour utiliser trois zones de disponibilité dans chaque région où il est déployé. Dans Azure, une [zone de disponibilité] est un datacenter autonome dans une région. Il dispose de sa propre alimentation, de son propre refroidissement et de sa propre connectivité réseau. Si vous inspectez un déploiement d'Azure Red Hat OpenShift, vous verrez un seul nœud de contrôle (machine virtuelle) et un seul nœud d'application dans chaque zone de disponibilité.

La *figure 4.1* montre comment les nœuds de contrôle et les nœuds d'application (calcul) sont répartis dans trois zones de disponibilité au sein d'une seule région Azure :

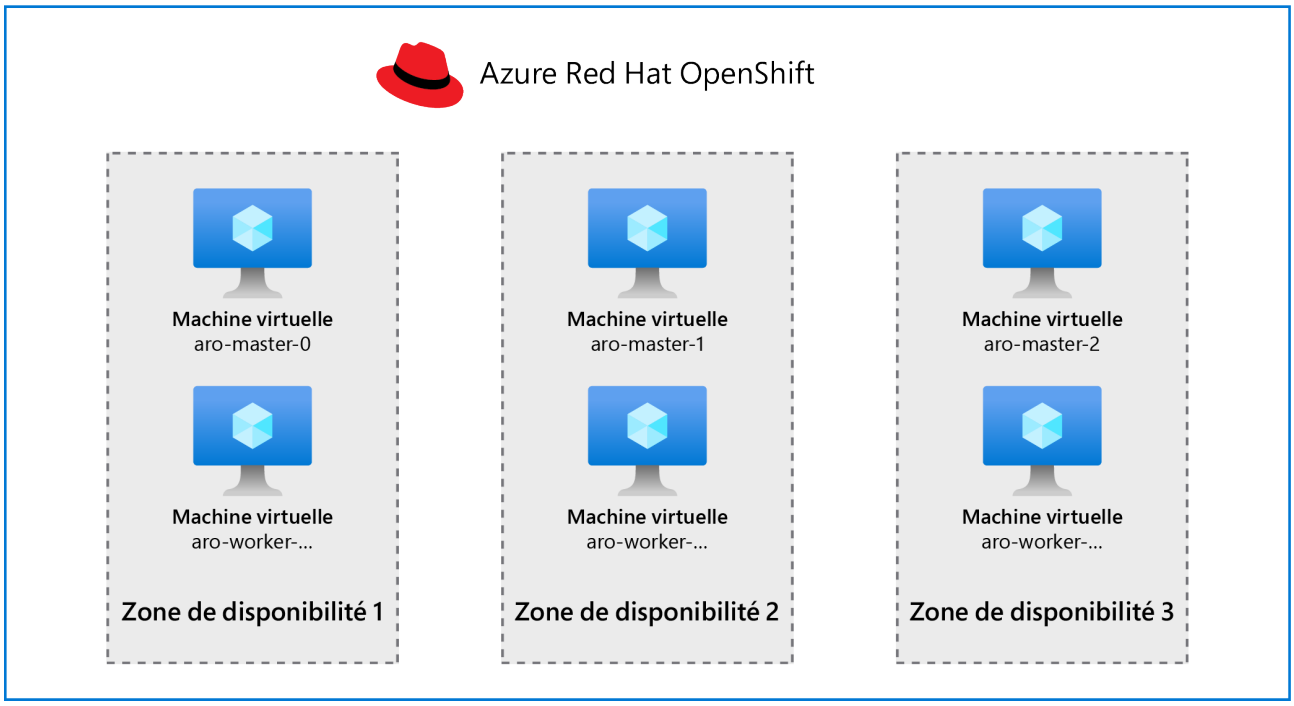


Figure 4.1 : Répartition des nœuds de contrôle et d'application dans trois zones de disponibilité au sein d'une seule région Azure

La conception du service Azure Red Hat OpenShift permet de le proposer comme un service entièrement géré et hautement disponible. Par conséquent, vous ne pouvez pas déployer moins de trois nœuds de contrôle et trois nœuds de calcul.

Concepts de réseau

Nous avons mentionné précédemment, dans la présentation d'Azure Red Hat OpenShift, cette excellente page de documentation sur les concepts de mise en réseau, disponible sur le site de documentation de Microsoft :

- Concepts de réseau pour Azure Red Hat OpenShift

Cette page présente une description détaillée de chaque composant réseau d'Azure Red Hat OpenShift : équilibreurs de charge, adresses IP publiques et privées, groupes de sécurité réseau, etc.

Voici quelques points essentiels que vous devez maîtriser :

- Vous pouvez déployer Azure Red Hat OpenShift sur un réseau virtuel existant ou sur un nouveau réseau virtuel. Seul un réseau virtuel est pris en charge. Plusieurs réseaux n'apporteraient aucun avantage supplémentaire, car OpenShift superpose son propre **réseau logiciel (SDN)**, appelé OVS.
- Taille minimale des sous-réseaux du nœud maître et du nœud d'application : `/27`.
- CIDR de pod par défaut : `10.128.0.0/14`.
- CIDR de service par défaut : `172.30.0.0/16`.
- Chaque nœud se voit attribuer un sous-réseau `/23` (512 adresses IP) pour ses pods. Vous ne pouvez pas modifier cette valeur.
- Les IP de sortie ne sont pas prises en charge pour le moment.
- Vous pouvez contrôler l'acheminement du trafic de sortie, notamment pour l'envoyer via le pare-feu Azure. Au moment de la rédaction de ce manuel, cette fonctionnalité est proposée en version préliminaire publique et documentée dans : [Contrôle du trafic de sortie].

Visibilité du réseau public ou privé

Vous entendez souvent dire que vous pouvez déployer Azure Red Hat OpenShift à la fois dans un déploiement public et privé. Même si c'est effectivement le cas, vous devez toutefois comprendre la différence entre rendre le plan de contrôle public/privé et rendre les applications de votre cluster publiques/privées.

Concernant la visibilité du serveur d'API, vous devez prendre cette décision au moment du provisionnement. Vous ne pouvez pas ajuster la visibilité publique/privée après le provisionnement du cluster.

Au moment de créer un cluster Azure Red Hat OpenShift ultérieurement dans ce chapitre, vous exécuterez la commande `az aro create`. Cette dernière accepte des arguments sur la visibilité du cluster. Exemple d'ajustement de la visibilité :

Deux serveurs privés

```
az aro create .... --apiserver-visibility Private --ingress-visibility Private
```

Serveur d'API privé et entrée de calcul publique

```
az aro create .... --apiserver-visibility Private --ingress-visibility Public
```

La visibilité de l'`apiserver` et de l'entrée des applications correspond au schéma de l'architecture Azure présenté sur la *figure 4.2*. Ce schéma montre comment Azure Red Hat OpenShift utilise les équilibreurs de charge Azure internes et publics où l'API et le routeur sont déployés en fonction des options de visibilité sélectionnées.

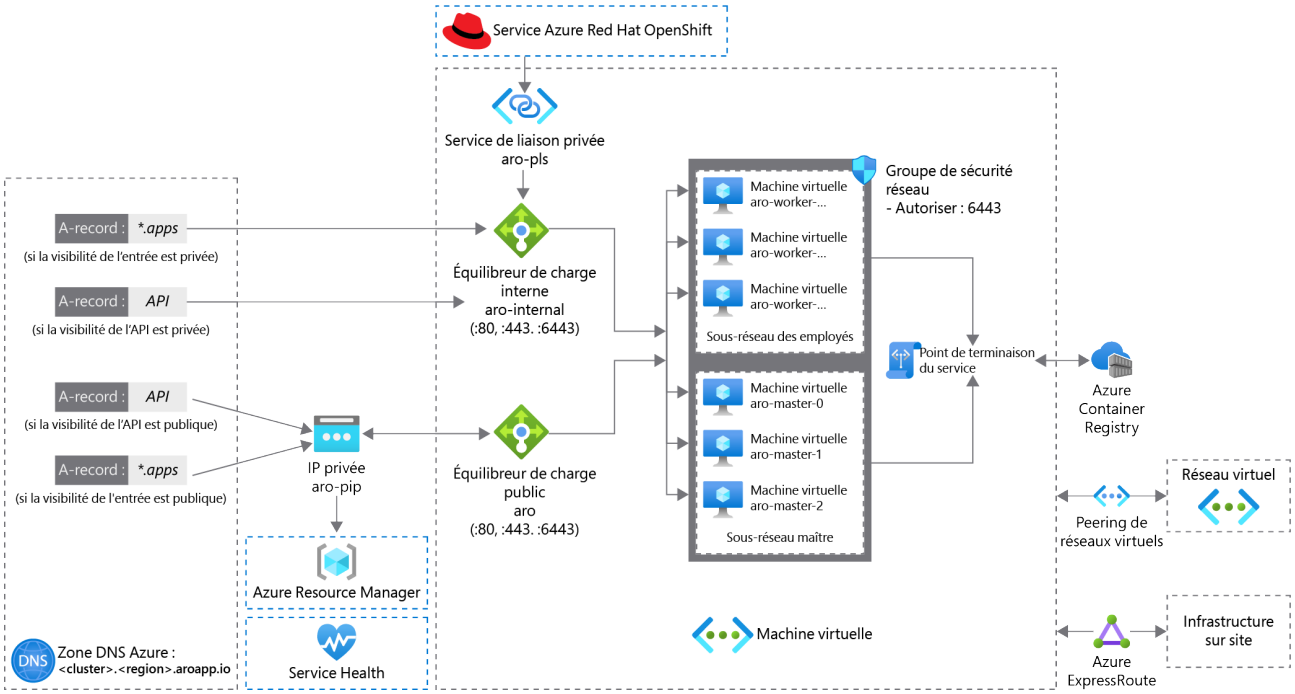


Figure 4.2 : Azure Red Hat OpenShift utilisant des équilibreurs de charge Azure internes et publics

Une description plus détaillée de la visibilité du serveur d'API et de l'entrée est présentée ci-après.

Visibilité du serveur d'API (plan de contrôle)

Vous pouvez définir `--apiserver-visibility` sur **public** ou **private** :

- **Private** (privé) signifie que le plan de contrôle de Red Hat OpenShift (historiquement appelé « nœuds maîtres »), où s'exécute l'API Kubernetes, n'est pas accessible depuis l'Internet public. Destiné aux développeurs et aux opérateurs pour contrôler le cluster, ce plan de contrôle peut être utilisé pour déployer, supprimer ou mettre à l'échelle des applications, ainsi que le cluster lui-même. Les entreprises qui disposent de connexions par routage express à Azure ou qui utilisent un **réseau privé virtuel (VPN)** pour accéder aux ressources informatiques devraient généralement choisir le mode privé.
- **Public** signifie que le plan de contrôle du cluster est accessible depuis l'Internet public. Malgré l'accès authentifié, le cluster reste sensible au risque d'attaque de tiers sur internet. Cependant, le réglage sur `public` est utile pour les environnements de laboratoire ou de test qui ne vous permettent pas de contrôler le réseau source de vos utilisateurs. Pour les environnements de stockage de données réelles ou tout type d'environnement de production, nous recommandons fortement de définir la visibilité du serveur d'API sur `private`.

Voici quelques exemples de cas où la connectivité du serveur d'API est nécessaire. Vous devez en tenir compte lors du choix entre public et privé :

- Les développeurs qui utilisent des scripts et des outils à partir de leur environnement de développement intégré (IDE) (par exemple, `kubectl rollout`)
- Les opérateurs qui inspectent l'état de leur cluster (par exemple, `kubectl get nodes`)
- Les serveurs CI/CD qui doivent inspecter ou ajuster l'état des déploiements (par exemple, Azure DevOps)
- Les outils de sécurité du cluster qui se connectent au cluster pour examiner l'état
- Les outils de surveillance qui s'appuient sur l'API Kubernetes

Dans la plupart des cas, vous pouvez configurer la mise en réseau pour utiliser des connexions **privées**. La liste ci-dessus n'est toutefois pas exhaustive, votre organisation pouvant inclure des exemples supplémentaires. Vous pouvez constater que la visibilité du serveur d'API **public** présente une exigence inattendue.

Visibilité entrante (applications)

La valeur de `--ingress-visibility` peut être « public » ou « private » :

- **Private** (privé) signifie que vous ne pouvez pas utiliser les services exposés (qui concernent les applications fonctionnant sur le cluster) pour vous connecter directement depuis l'Internet public. Vous pouvez toujours configurer le routage réseau pour permettre aux utilisateurs de se connecter à vos applications via le pare-feu Azure ou un pare-feu d'application web, éventuellement exécuté sur un réseau Azure distinct. Le paramètre **private** convient parfaitement si votre organisation envisage uniquement d'utiliser les applications du cluster en interne, par exemple, pour le traitement des salaires, l'analyse des données ou d'autres applications web internes.
- **Public** signifie que les applications de votre cluster sont accessibles depuis l'Internet public. Vous devez toutefois configurer une ressource d'entrée ou de routage pour permettre l'accès à ces applications. C'est le cas si vous hébergez un site web public de commerce électronique ou une autre application publique sur Red Hat OpenShift.

L'entrée porte parfois le nom de « routeur » d'OpenShift.

Recommandations pour la production

Fortes recommandations :

- Accédez au cluster via une connexion privée et non via l'Internet public. Azure ExpressRoute est la meilleure option lorsque le cluster requiert une connectivité permanente à partir d'un réseau sur site ou d'un siège social. Sinon, un VPN dans Azure permet également de fournir une connexion privée. Vous trouverez plus de détails à ce sujet dans la section **Connectivité hybride**.
- Définissez la visibilité du serveur d'API (plan de contrôle) sur `private` et limitez éventuellement davantage l'accès à l'aide de pare-feu.
- Définissez la visibilité entrante (applications) des applications exécutées au sein de votre organisation sur `private`. Si vous devez héberger un cas d'utilisation des applications sur Azure Red Hat OpenShift sur l'Internet public, configurez un cluster Azure Red Hat OpenShift distinct (au moins un cluster pour les applications internes et un autre pour les applications externes) en définissant la visibilité entrante sur public. Vous pouvez toutefois mélanger des entrées `public` et `private` au sein d'un même cluster.

Naturellement, la plupart des organisations consulteront leurs équipes de sécurité et de mise en réseau d'Azure pour déterminer les contrôles supplémentaires éventuellement requis. Par exemple, certaines organisations exigent que les applications web soient protégées par un pare-feu dédié. Ce modèle de déploiement est également usuel lors du déploiement d'applications sur Azure Red Hat OpenShift.

Connectivité hybride

La plupart des organisations qui déploient Azure Red Hat OpenShift exécutent des applications qui doivent se connecter à des services de prise en charge fonctionnant dans des environnements sur site. La connexion d'Azure à un environnement sur site implique une connectivité hybride ou une architecture de cloud hybride. Vous pouvez rétablir la connectivité avec les environnements sur site de plusieurs façons. Les deux options les plus notables sont les suivantes :

- **VPN** : convient pour une connectivité peu complexe à Azure. En règle générale, la connexion des VPN passe par la passerelle VPN Azure. Pour en savoir plus, consultez Qu'est-ce que la passerelle VPN Azure ?
- **Circuits Azure ExpressRoute** : conviennent pour une connexion dédiée, robuste et permanente à Azure. Pour en savoir plus, consultez Qu'est-ce qu'Azure ExpressRoute ?

Indépendamment de la méthode de connexion utilisée, vous pouvez utiliser ces deux solutions pour connecter les applications sur site aux applications exécutées sur Azure Red Hat OpenShift, et inversement.

Lors de la connexion d'un environnement sur site à Azure Red Hat OpenShift, l'utilisation d'un réseau virtuel de type hub pour se connecter est fréquente. La *figure 4.3* présente un schéma qui explique le fonctionnement de la connectivité avec Azure ExpressRoute :

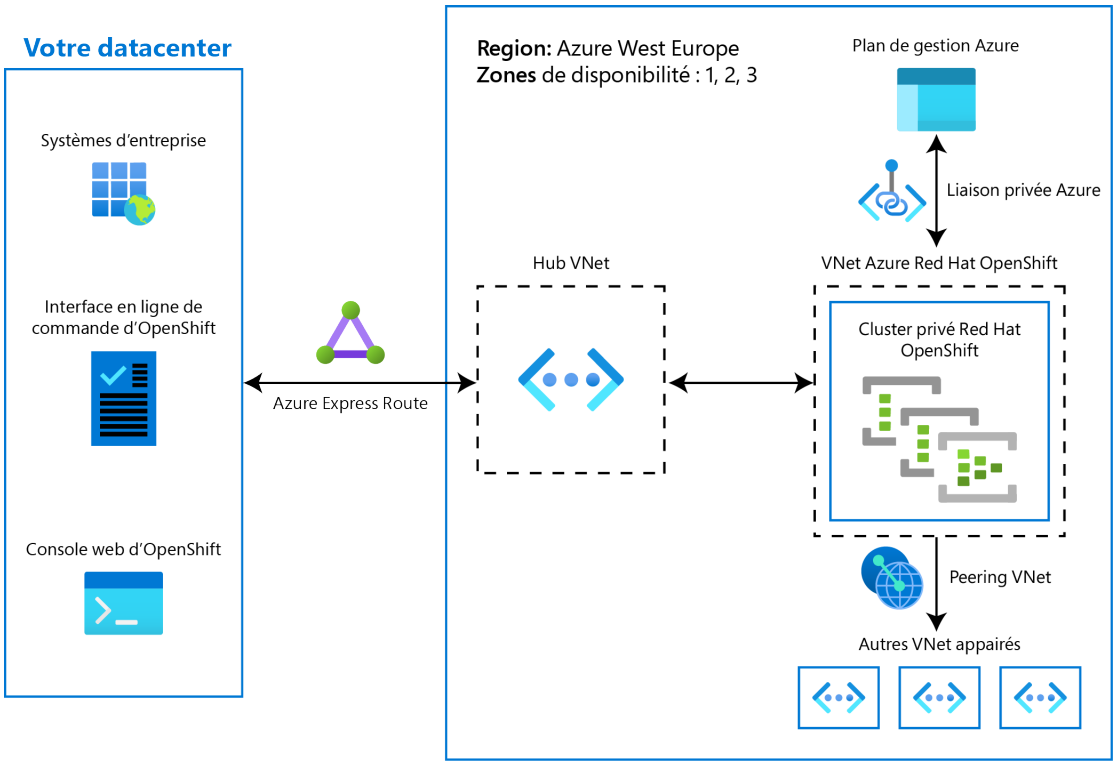


Figure 4.3 : Connectivité avec Azure ExpressRoute

Le schéma précédent montre l'architecture globale d'Azure ExpressRoute se connectant à Azure Red Hat OpenShift. Bien que ce schéma montre Azure ExpressRoute, la connectivité via un VPN est très similaire d'un point de vue conceptuel.

Considérations relatives aux applications fonctionnant dans une architecture hybride

Lors de l'exécution d'applications sur Azure Red Hat OpenShift qui se connectent à des environnements sur site, les propriétaires d'applications doivent tenir compte de plusieurs éléments :

- **Latence de la connexion :** alors que la latence de la connexion d'Azure ExpressRoute vers les environnements sur site est relativement faible, les réseaux VPN qui passent par l'Internet public peuvent présenter une latence considérablement plus élevée. Pour certaines applications, comme les serveurs web, où la connexion ne fait que transmettre le trafic HTTP, il est peu probable que cela pose des problèmes. Toutefois, si une application côté serveur fonctionnant sur ce serveur web doit se connecter à une base de données fonctionnant sur site, l'impact sur les performances peut être considérable.
- **Coût du trafic d'entrée/sortie :** certains types de connexion facturent le trafic d'entrée et de sortie, soit via Azure, soit par le fournisseur de connexion. À titre de précaution, vous pouvez commencer par mesurer les coûts dans un environnement de test, puis évaluer leur niveau en cas de charge maximale. De cette manière, vous ne serez pas surpris.
- **Scénario de défaillance :** alors que les connexions Azure ExpressRoute sont conçues pour être permanentes et robustes, les connexions VPN sont souvent sujettes à des interruptions ou à des connexions/déconnexions fréquentes. Comme elles utilisent également l'Internet public, la latence et la qualité de service des connexions VPN peuvent varier assez fréquemment au cours de la journée. Vous devez tester les performances de l'application dans des conditions de liaison hybride médiocres et lorsque la connexion fonctionne de manière optimale. De même, en cas d'interruption de la connexion pendant plusieurs heures, l'application exécutée sur Azure Red Hat OpenShift peut-elle continuer à fonctionner de manière dégradée ou sa disponibilité dépend-elle entièrement de la connexion hybride ?

Bien que les éléments évoqués suffisent normalement pour déterminer le fonctionnement d'une architecture hybride au sein de votre organisation, cette dernière peut être amenée à devoir satisfaire d'autres critères figurant sur vos propres listes de contrôle internes.

Synthèse

Ce chapitre a traité plusieurs questions de pré-provisionnement usuelles que vous devez vous poser et résoudre avant de déployer Azure Red Hat OpenShift. Le chapitre suivant fournit des conseils utiles sur les ressources lors du déploiement réel du cluster.

Chapitre 5

Provisionnement d'un cluster Azure Red Hat OpenShift

Voici un récapitulatif des points passés en revue. Dans ce manuel, nous avons déjà abordé les points suivants :

- Le *chapitre 2*, *Présentation de Red Hat OpenShift*, présente brièvement Red Hat OpenShift et explique les avantages de choisir OpenShift plutôt que Kubernetes nu.
- Le *chapitre 3*, *Azure Red Hat OpenShift*, décrit les spécificités du service cloud géré Azure Red Hat OpenShift, ainsi que les concepts clés, notamment son architecture, sa gestion, son authentification, son assistance et ses considérations tarifaires.
- Le *chapitre 4*, *Pré-provisionnement – Questions sur l'architecture d'entreprise*, aborde les questions courantes auxquelles les organisations doivent répondre avant de déployer Azure Red Hat OpenShift.

Nous sommes désormais prêts à provisionner et à déployer un cluster. Vous trouverez les instructions officielles de déploiement sur le site de documentation (le provisionnement fait partie du déploiement ; provisionner un déploiement signifie s'assurer que l'infrastructure informatique nécessaire pour supporter un déploiement est en place).

[Didacticiel – Création d'un cluster Azure Red Hat OpenShift 4]

Ce chapitre ne couvre pas les commandes individuelles que vous devez saisir pour créer un cluster. En effet, les commandes changent au fil du temps et la documentation couvre déjà ce sujet en détail.

Ce chapitre fournira cependant des conseils sur le processus global et les éléments à prendre en compte lors du déploiement d'un cluster.

Déploiements manuels – Calendrier prévisionnel

Pour certaines organisations, la maîtrise du calendrier de provisionnement d'un cluster est importante. Voyons cela en détail.

Voici un processus global des conditions préalables au déploiement :

- Ligne de commande az déployée sur la station de travail d'un administrateur système
- Fournisseurs de ressources enregistrés à utiliser avec votre souscription Azure
- Secret d'extraction Red Hat (facultatif)
- Domaine pour votre cluster (facultatif)
- Réseau virtuel incluant deux sous-réseaux vides, l'un pour les nœuds du plan de contrôle et l'autre pour les nœuds d'application.

Concernant le temps nécessaire à la mise en place de ces conditions préalables, un administrateur Azure et Red Hat OpenShift compétent pourrait probablement accomplir ces tâches en 30 minutes la première fois ; en 10 minutes seulement s'il exécute ces étapes de manière répétée dans le cadre d'un processus manuel.

Une fois les conditions préalables remplies, le processus de provisionnement automatisé dure généralement entre 25 et 40 minutes, selon l'activité au sein de la région Azure.

Automatisation du déploiement

Comme elle sait que le processus de provisionnement d'Azure Red Hat OpenShift est automatisé, une organisation essaie normalement d'automatiser les étapes préalables au moyen d'outils : création de réseaux, de sous-réseaux, de principes de service, etc.

De nombreux outils permettent d'automatiser ces étapes. Voici quelques outils recommandés :

- Outil de ligne de commande az : lorsqu'il est automatisé, cet outil est normalement installé dans un conteneur, ou élément similaire, dans le cadre d'un processus CI/CD. Les outils généralement utilisés comprennent Jenkins, Azure DevOps, voire Ansible. Notez que vous ne devez déployer l'outil en ligne de commande az qu'une seule fois. Toutefois, vous devrez peut-être définir l'ID de souscription Azure des clusters supplémentaires pour refléter les différentes parties de l'organisation.
- Fournisseurs de ressources enregistrés à utiliser avec votre souscription Azure : comme précédemment, ils font partie de la configuration de l'outil en ligne de commande az.
- Secret d'extraction Red Hat (facultatif) : Red Hat dispose d'une API REST documentée et prise en charge pour obtenir un secret d'extraction, dont les informations sont disponibles dans cet **article**.
- Domaine pour votre cluster (facultatif) : tout dépend de la façon dont vous créez les enregistrements DNS. Si vous utilisez Azure DNS, Terraform, Ansible ou d'autres outils d'automatisation Azure populaires peuvent le faire pour vous.
- Réseau virtuel intégrant deux sous-réseaux vides, un pour les nœuds du plan de contrôle (maître) et un pour les nœuds d'application (calcul) : les outils d'automatisation Azure les plus courants permettent généralement d'automatiser cette opération (Terraform, Ansible ou des outils similaires peuvent créer ce réseau et ces sous-réseaux pour vous).

L'automatisation des étapes préalables permet concrètement de réduire la durée d'exécution, qui est de 30 ou 10 minutes dans le cadre d'un processus manuel, à seulement une ou deux minutes. L'accélération du déploiement du cluster (qui requiert toujours entre 25 et 40 minutes) est impossible. Cependant, le processus de déploiement de bout en bout peut s'avérer très rapide par rapport au déploiement sur site.

L'automatisation du déploiement présente de nombreux avantages qui vont au-delà de la simple accélération du processus : les clients qui y ont recours tirent profit de la mise en place d'un processus robuste et reproductible, facilement consigné et audité. Les clients intègrent également très fréquemment des éléments de catalogue en libre-service à leurs propres portails. De cette manière, les équipes peuvent provisionner et déprovisionner facilement les clusters Azure Red Hat OpenShift sans aucune interaction de la part de l'équipe en charge de la plateforme cloud.

Accès au cluster

Cette section explique simplement comment accéder à votre cluster Azure Red Hat OpenShift après son provisionnement.

Via l'interface web utilisateur

À partir d'un shell Bash si vous avez installé l'interface en ligne de commande, ou depuis une session d'Azure Cloud Shell (Bash) dans votre portail Azure, récupérez l'URL de connexion de votre cluster en exécutant la commande :

```
az aro show -n $CLUSTER_NAME -g $RG_NAME --query "consoleProfile.url" -o tsv
```

Vous devez obtenir une URL ressemblant à `openshift.xxxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`. L'URL de connexion de votre cluster sera `https://` suivi de la valeur `consoleProfile.url` ; par exemple, `https://openshift.xxxxxxxxxxxxxxxxxxxx.eastus.aroapp.io`.

Ouvrez cette URL dans votre navigateur. Vous serez invité à vous connecter avec l'utilisateur `kubeadmin`. Utilisez le nom d'utilisateur et le mot de passe que l'installateur vous a fournis au cours du processus d'installation.

Une fois connecté, la console web Azure Red Hat OpenShift devrait s'afficher.

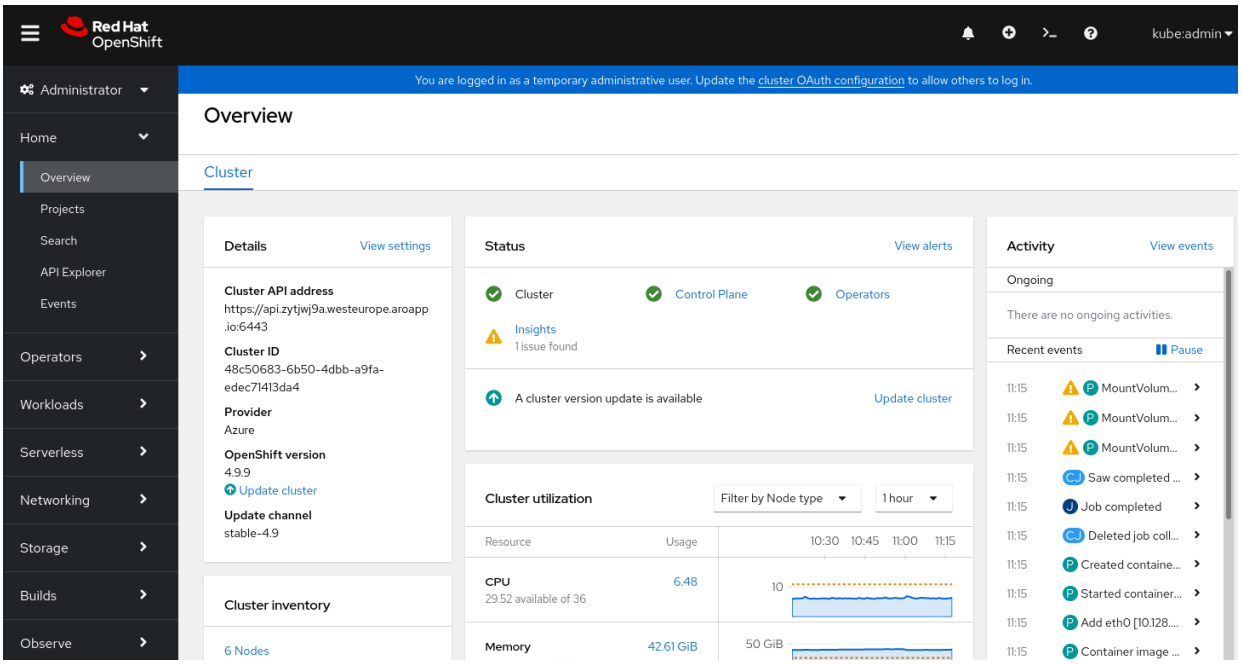


Figure 5.1 : Console web Azure Red Hat OpenShift

Prenez le temps d'explorer la console web. Notez qu'elle doit exécuter une version récente d'OpenShift et que l'état de tous les composants doit être sain ou le devenir après l'installation.

Via l'interface en ligne de commande OpenShift (oc)

Vous devez télécharger la dernière version de l'interface en ligne de commande OpenShift (oc). Pour ce faire, connectez-vous simplement et consultez la page https://console.redhat.com/openshift/downloads.

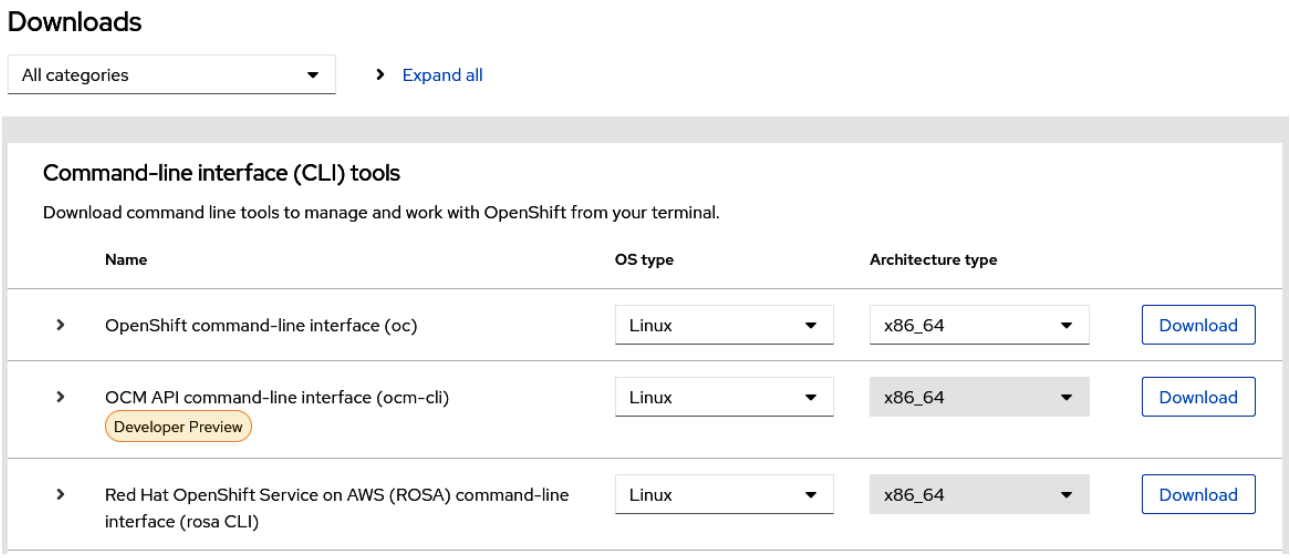


Figure 5.2 : Téléchargement de l'interface en ligne de commande OpenShift

Extrayez l'archive (`.tar.gz` ou `.zip`) sur votre système, puis insérez la commande `oc` dans le chemin d'accès. Sous Linux, il est tout à fait normal d'ajouter la commande `oc` dans `/usr/local/sbin/`.

Exécution de l'interface en ligne de commande OpenShift et connexion à votre cluster

Pour vous authentifier auprès de votre cluster à partir de la ligne de commande, vous devrez récupérer la commande de connexion et le jeton de la console web. Connectez-vous à la console web avec le navigateur, cliquez sur le nom d'utilisateur en haut à droite, puis cliquez sur **Copy login command** (Copier la commande de connexion).

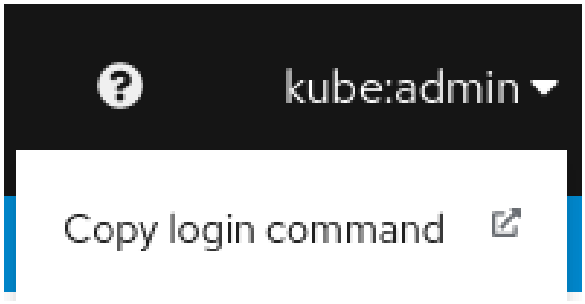


Figure 5.3 : Copier la commande de connexion

Une nouvelle page similaire à l'exemple ci-après s'affiche :

Figure 5.4 : Connexion avec le jeton d'API

Copiez ensuite cette commande et collez-la dans un terminal pour vous connecter à votre cluster Azure Red Hat OpenShift.

Exemple : si vous utilisez la session Azure Cloud Shell (Bash) dans votre portail Azure, en collant cette commande de connexion, vous obtenez la commande `oc status` :

```
user@Azure: oc status
In project default on server https://api.cyki1k6g.westeurope.aroapp.io:6443

svc/openshift - kubernetes.default.svc.cluster.local
svc/kubernetes - 172.30.0.1:443 -> 6443

Affichez les détails avec 'oc describe <resource>/<name>' ou répertoriez tout avec 'oc get all'.
```

Prenez à présent le temps d'étudier davantage la ligne de commande `oc` et de vous familiariser avec votre nouvel environnement Azure Red Hat OpenShift. Si vous êtes déjà un utilisateur expérimenté d'OpenShift 4, ce service cloud d'Azure Red Hat OpenShift devrait vous sembler très familier. Il devrait se comporter exactement de la même manière que les autres environnements OpenShift 4 que vous avez pu utiliser précédemment.

Synthèse

Ce chapitre est court, car les instructions officielles de provisionnement sont correctement mises à jour. Elles constituent le manuel définitif d'utilisation d'Azure Red Hat OpenShift dans le cadre de votre souscription Azure. Vous remarquerez que les instructions de provisionnement sont considérablement plus simples que celles utilisées dans un environnement OpenShift autogéré. En effet, le service Azure Red Hat OpenShift a bénéficié d'un gros travail technique pour assurer une intégration étroite à Azure et le déploiement d'une architecture prescriptive adaptée à toutes les situations, correctement testée et bien prise en charge. Dans l'ensemble, il s'agit d'un avantage pour les organisations, qui réduisent ainsi la durée de provisionnement d'Azure Red Hat OpenShift au profit du déploiement des applications.

Chapitre 6

Post-provisionnement – Jour 2

Après le déploiement d'Azure Red Hat OpenShift et pour préparer un cluster pour la production, vous devez généralement exécuter quelques activités de post-provisionnement. Ce chapitre couvre la plupart des activités standard de post-provisionnement :

- Authentification – Azure Active Directory : y compris comment synchroniser les groupes d'utilisateurs avec un opérateur communautaire
- Opérateurs : dont OperatorHub
- Compréhension de la journalisation : y compris le transfert des journaux
- Compréhension de la surveillance : y compris la surveillance des conteneurs OpenShift
- Mises à jour et correctifs : y compris le cycle de vie des versions prises en charge
- Mise à l'échelle du cluster : y compris la mise à l'échelle manuelle et automatique du cluster
- Mise à l'échelle de l'application : brève description de la mise à l'échelle des applications
- Configuration d'un secret d'extraction : enregistrement dans OpenShift Cluster Manager
- Plages de limites : y compris où une plage de limites peut être appliquée
- Stockage persistant : y compris les classes de stockage disponibles et prises en charge
- Sécurité et conformité : remarque concernant les contrôles de sécurité

Chacune de ces sections détaille différentes tâches de post-provisionnement pour faciliter votre compréhension de la préparation d'un cluster fraîchement déployé en vue de l'utiliser dans les applications de production.

Authentification – Azure Active Directory

Azure Red Hat OpenShift prend en charge tous les fournisseurs d'authentification répertoriés dans la documentation d'OpenShift. Consultez la liste complète des fournisseurs d'authentification. Toutefois, la majorité des clients sont susceptibles d'utiliser Azure Active Directory, qui est déjà disponible sur Azure, pour fournir une authentification et une authentification unique et unifiée à Azure Red Hat OpenShift.

La configuration de l'intégration d'Azure Active Directory est délibérément une opération du « jour 2 », car dans certaines situations, les organisations souhaitent utiliser un autre fournisseur de configuration. Le processus de configuration et d'installation d'Azure Active Directory prend environ 15 minutes la première fois, puis, une fois familiarisé avec le processus, vous constaterez qu'il ne vous faudra que cinq minutes. De nombreuses organisations choisissent d'automatiser certaines parties de ce provisionnement en utilisant des outils tels que les modèles ARM ou les playbooks Ansible.

- Configurer Azure Active Directory pour Azure Red Hat OpenShift (portail graphique)
- Configurer Azure Active Directory pour Azure Red Hat OpenShift (interface en ligne de commande)

Utilisation des groupes d'utilisateurs Active Directory

La configuration de l'authentification Azure Active Directory ne permettra aux utilisateurs de se connecter qu'avec les informations d'identification existantes. Elle ne transfère pas les groupes d'utilisateurs existants d'un utilisateur vers Red Hat OpenShift. Il est assez courant qu'une organisation veuille importer des groupes d'utilisateurs depuis Active Directory afin de les utiliser pour configurer les autorisations des utilisateurs dans OpenShift. Pour ce faire, un opérateur distinct, soutenu par la communauté, est disponible : l'opérateur de synchronisation de groupe.

- Opérateur de synchronisation de groupe sur GitHub

Notez que l'opérateur de synchronisation de groupe est pris en charge par la communauté, ce qui signifie qu'il n'est pas officiellement pris en charge par Red Hat ou Microsoft au moment de la rédaction de ce document. Ce manuel recommande de suivre les instructions détaillées de configuration et d'installation de l'opérateur accompagnant le fichier README du projet.

Opérateurs

Les opérateurs dans OpenShift 4 représentent l'un des composants fondamentaux de la plateforme et apportent une valeur ajoutée considérable aux utilisateurs de Red Hat OpenShift. Les opérateurs reposent sur du code et exécutent un service dans un conteneur du cluster. Certains opérateurs sont responsables de la maintenance d'éléments tels que la mise en réseau des clusters, la maintenance de la configuration des machines et les mises à niveau des clusters. Ceux-ci sont souvent appelés opérateurs de cluster. Vous en trouverez la liste dans la section **Administration** de la console OpenShift.

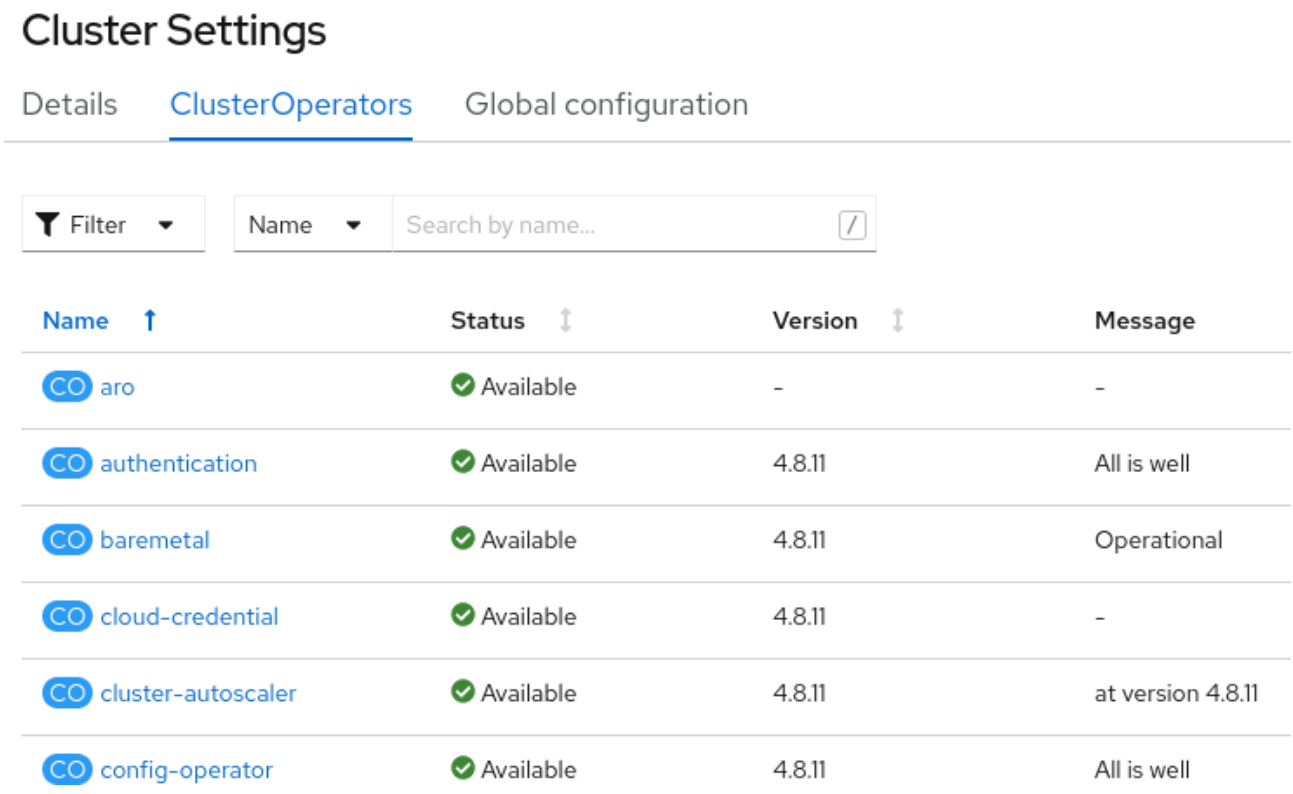


Figure 6.1 : Paramètres du cluster

La capture d'écran précédente indique même la présence d'un opérateur dédié uniquement à Azure Red Hat OpenShift, en charge de la maintenance des parties du service et du maintien du cluster dans un état de configuration pris en charge.

Les opérateurs peuvent assurer la maintenance de nombreux paramètres, dont l'intégrité d'un service, les mises à jour, les correctifs, la mise à l'échelle et plusieurs autres fonctions.

OperatorHub

Au-delà des opérateurs de cluster standard qui s'exécutent en arrière-plan sur chaque cluster, les administrateurs ont accès à de nombreux autres opérateurs via OperatorHub. OperatorHub permet de rechercher facilement des opérateurs communautaires et d'entreprise populaires qu'un administrateur pourrait vouloir rendre disponibles sur l'installation Red Hat OpenShift. OperatorHub est disponible dans la barre latérale de chaque cluster OpenShift.

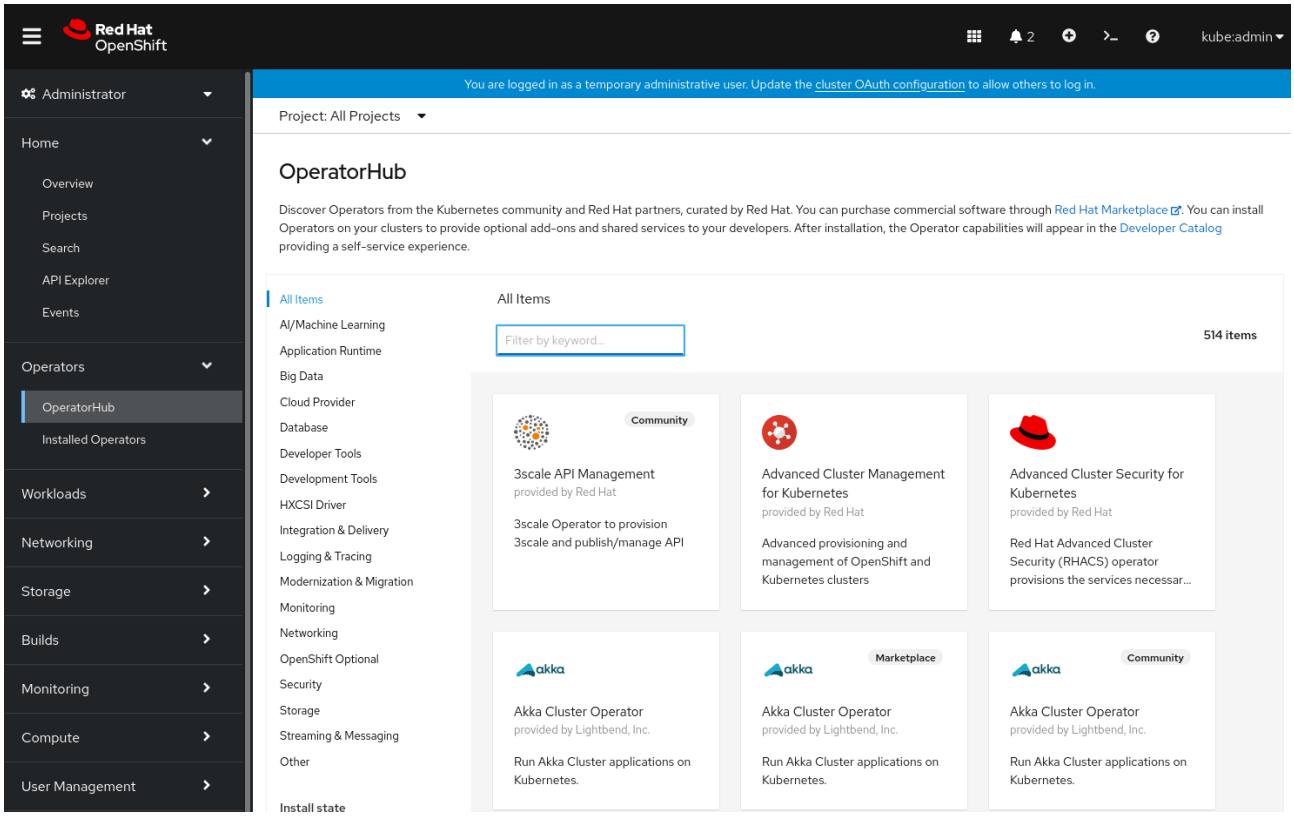


Figure 6.2 : OperatorHub

Grâce aux opérateurs, les administrateurs peuvent déployer et assurer la maintenance des logiciels courants rapidement et facilement, sans se soucier de la configuration et du code de Kubernetes. De nombreux produits Red Hat sont désormais proposés sous forme d'opérateurs, tels que Advanced Cluster Management, Red Hat OpenShiftService Mesh et Red Hat OpenShift Pipelines. Cependant, une importante bibliothèque d'opérateurs est également disponible pour les logiciels non Red Hat, comme la base de données MariaDB, Couchbase ou les pilotes de stockage en mode bloc IBM.

Nous vous recommandons de consulter les liens suivants pour en savoir plus sur les opérateurs :

- Operatorhub.io
- Opérateurs dans OpenShift

Compréhension de la journalisation

Azure Red Hat OpenShift utilise la même architecture de journalisation et de surveillance que Red Hat OpenShift. Les deux offres utilisent les mêmes opérateurs pour la journalisation.

Les journaux sont répartis en trois catégories :

- **Application :** journaux de conteneurs générés par les applications utilisateur exécutées dans le cluster, à l'exception des applications de conteneurs d'infrastructure.
- **Infrastructure :** journaux générés par les composants d'infrastructure exécutés sur le cluster et les nœuds OpenShift Container Platform, tels que les enregistrements de journaux. Les composants d'infrastructure sont des pods qui fonctionnent dans les projets `openshift*`, `kube*` ou par défaut.
- **Audit :** journaux générés par auditd, le système d'audit des nœuds, et stockés dans le fichier `/var/log/audit/audit.log`, et les journaux d'audit du serveur d'API Kubernetes et du serveur d'API OpenShift.

Vous trouverez une description plus détaillée de la journalisation d'OpenShift dans la section sur la compréhension de la journalisation de Red Hat OpenShift de la documentation du produit.

Utilisation du transfert des journaux de cluster vers Azure Monitor

Une intégration courante consiste à envoyer les journaux d'Azure Red Hat OpenShift au service Container insights d'Azure Monitor. Ce service s'appelle parfois Azure Log Analytics. Il permet une conservation persistante et peu coûteuse des journaux dans Azure.

L'architecture de journalisation d'OpenShift exécute Fluent Bit sur chaque nœud. La première approche d'un opérateur pourrait être d'ajuster la configuration de ce service pour envoyer les journaux directement. Toutefois, l'ajustement de la configuration de la journalisation dans Azure Red Hat OpenShift ne relève pas de la politique d'assistance. OpenShift dispose d'un mécanisme intégré, appelé `ClusterLogForwarder`, qui permet de transférer les logs sans toucher à la prise en charge.

Avec `ClusterLogForwarder`, vous pouvez transférer les journaux vers Elasticsearch, Flutentd et syslog. Cependant, Azure Monitor ne prend pas directement en charge ces protocoles. Vous pouvez appliquer une solution de contournement via une instance intermédiaire supplémentaire de Fluent Bit, qui reçoit les journaux d'OpenShift et les transmet à Azure Monitor. Cette approche est décrite dans la documentation de Microsoft, ainsi que dans la documentation de la communauté.

Compréhension de la surveillance

Un service géré ne requiert généralement pas la mise en œuvre d'une surveillance personnalisée complexe d'Azure Red Hat OpenShift, car celle-ci est fournie dans le cadre de votre service payant.

La volonté de surveiller les conteneurs OpenShift à l'aide d'Azure Container Insights est toutefois très fréquente. Cette fonctionnalité proposée en version préliminaire publique est décrite ici.

Mises à jour et correctifs

Lorsque vous provisionnez un cluster Azure Red Hat OpenShift, aucun canal de mise à jour n'est sélectionné. Cela signifie que votre cluster ne recevra pas de mises à jour par défaut.

Pour afficher votre canal de mise à jour, accédez à **Administration** → **Cluster Settings** (Paramètres du cluster) dans la barre latérale de navigation. Voici une vue des paramètres du cluster peu après le provisionnement d'un cluster Azure Red Hat OpenShift :

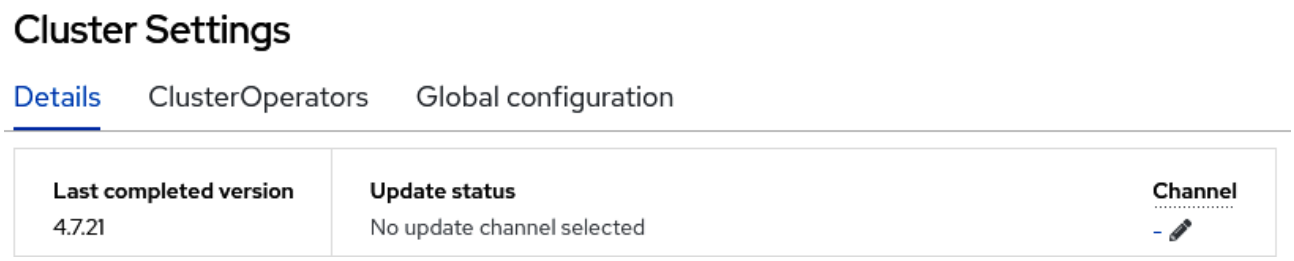


Figure 6.3 : Paramètres du cluster après son provisionnement

Pour sélectionner un canal de mise à jour, sélectionnez le lien « - ». Les options disponibles s'affichent :

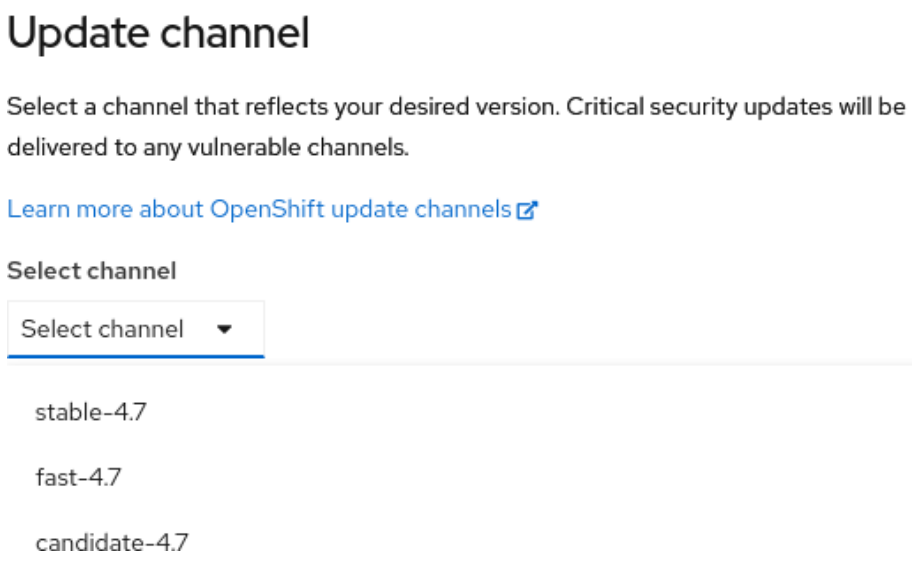


Figure 6.4 : Sélection d'un canal de mise à jour

Pour comprendre les différences entre `stable`, `fast` et `candidate`, consultez la page de documentation sur la mise à niveau des canaux et des versions.

Il est vivement recommandé que tous les clusters en production utilisent un canal **stable**.

Cycle de vie des versions prises en charge

Vous devez connaître les versions d'Azure Red Hat OpenShift prises en charge pour planifier votre déploiement de production. La page sur le cycle de vie formel contient des informations permettant aux organisations de déterminer les versions prises en charge et non prises en charge.

Page du cycle de vie du support pour Azure Red Hat OpenShift

Informations simplifiées du contenu de cette page :

Azure Red Hat OpenShift prend en charge deux versions mineures **généralement disponibles** de Red Hat OpenShift Container Platform :

- La dernière version mineure généralement disponible sortie dans Azure Red Hat OpenShift (la version N)
- Une version mineure antérieure (N-1)

Red Hat OpenShift Container Platform utilise la création de versions sémantiques. La création de versions sémantiques utilise différents niveaux de numéros de version pour spécifier différents niveaux de versions. Le tableau suivant illustre les différentes parties d'un numéro de version sémantique en prenant comme exemple le numéro de version 4.9.3 :

Version majeure (x)	Version mineure (y)	Correctif (z)
4	9	3

Chaque chiffre de la version indique la compatibilité générale avec la version précédente :

- **Version majeure :** aucune version majeure n'est prévue pour l'instant. Les versions majeures changent en présence de modifications d'API incompatibles ou d'une rétrocompatibilité potentiellement rompue.
- **Version mineure :** publication tous les trois mois environ. Les mises à jour mineures peuvent inclure des ajouts de fonctionnalités, des améliorations, des dépréciations, des suppressions, des corrections de bugs et des améliorations de la sécurité.
- **Correctifs :** publication chaque semaine (en général) ou au besoin. Les mises à niveau de la version des correctifs peuvent inclure des corrections de bugs et des améliorations de la sécurité.

Pour mieux déterminer les versions prises en charge, consultez la page sur le cycle de vie du support pour Azure Red Hat OpenShift.

Mise à l'échelle du cluster

Red Hat OpenShift Container Platform et, par extension, Azure Red Hat OpenShift, reposent sur une architecture évolutive. Lors de la planification de la mise à l'échelle dans le contexte d'OpenShift, les administrateurs et les propriétaires d'applications doivent généralement considérer la mise à l'échelle du cluster et la mise à l'échelle des applications comme deux objets distincts.

Cette section couvre la mise à l'échelle des clusters. La mise à l'échelle des applications fait l'objet d'une courte présentation dans une section distincte.

Prise en charge maximale

La mise à l'échelle d'un cluster consiste à ajouter des nœuds de calcul supplémentaires au cluster pour augmenter la capacité de calcul des applications exécutées dans le cluster. Naturellement, vous devez vous demander si vous devez également mettre à l'échelle le plan de contrôle. Azure Red Hat OpenShift déploie déjà trois nœuds de plan de contrôle, qui, en principe, ont une capacité suffisante pour la mise à l'échelle en fonction du nombre maximal de nœuds de calcul pris en charge dans un cluster Azure Red Hat OpenShift (actuellement 60).

Au moment de la rédaction du présent document, trois tailles d'instances de nœuds de plan de contrôle sont prises en charge : `Standard_D8s_v3`, `Standard_D16s_v3` et `Standard_D16s_v3`.

Les types d'instances Azure pris en charge pour les nœuds de calcul sont plus nombreux : optimisés pour le calcul (série F), optimisés pour la mémoire (série E) et polyvalents (série D).

Une liste des types d'instances Azure actuellement pris en charge pour Azure Red Hat OpenShift est disponible sur la page présentant la **stratégie de prise en charge**.

Déploiement minimal et remise à zéro

Les organisations souhaitent parfois déployer des clusters plus petits à des fins de test et de développement, ou à d'autres fins qui acceptent une disponibilité réduite. Actuellement, la taille minimale d'un cluster comprend trois nœuds de plan de contrôle et trois nœuds d'application. Aucune mise à l'échelle inférieure n'est prise en charge.

Mise à l'échelle manuelle du cluster

Les opérateurs qui consultent le portail Azure pourraient se laisser tenter par l'ajout de nœuds de calcul supplémentaires au cluster Azure Red Hat OpenShift en créant directement une machine virtuelle Azure. Cela est toutefois impossible, car le groupe de ressources qui contient Azure Red Hat OpenShift est « verrouillé » du point de vue de l'administrateur Azure. C'est le cas quelles que soient les autorisations ; même les utilisateurs disposant de l'autorisation de niveau **propriétaire** ne peuvent pas modifier le contenu d'un groupe de ressources Azure Red Hat OpenShift. Par conséquent, des erreurs d'autorisation refusée apparaîtront si vous tentez de créer une machine virtuelle manuellement dans le groupe de ressources Azure Red Hat OpenShift.

Le mécanisme de mise à l'échelle d'Azure Red Hat OpenShift prévu comprend l'utilisation de la fonction `MachineSets` d'OpenShift qui permet à OpenShift de déployer un nouveau nœud d'application à la demande. Les utilisateurs ayant des privilèges d'administrateur de cluster peuvent voir **MachineSets** sous **Compute** (Calculer).

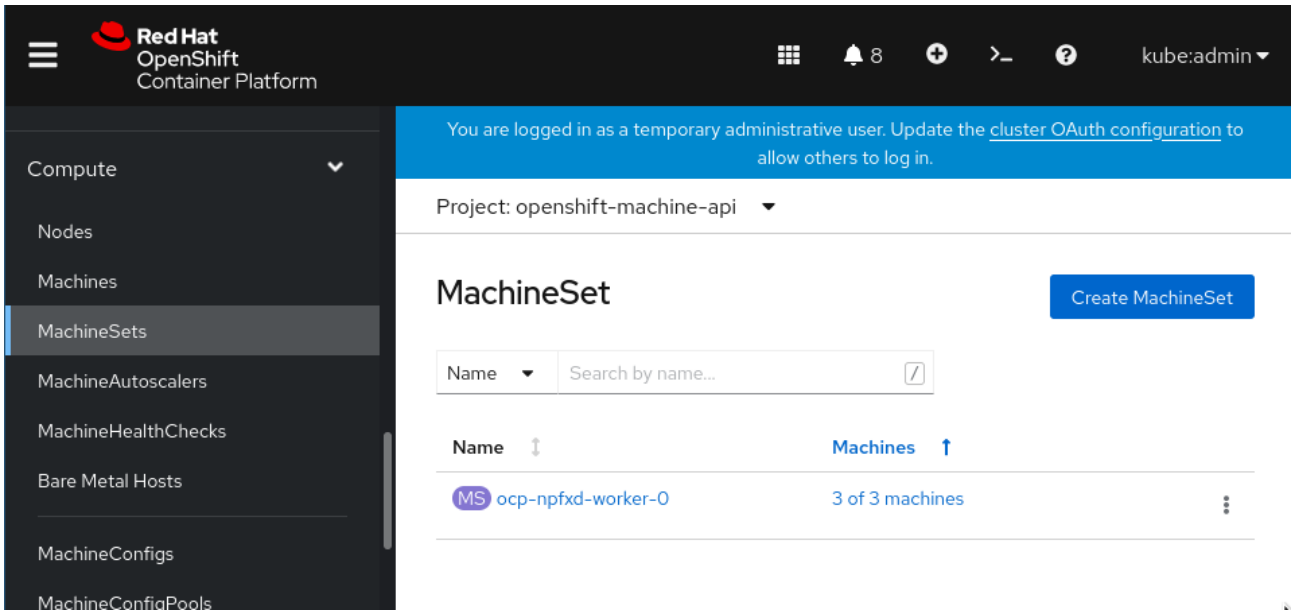


Figure 6.5 : Accès administrateur à MachineSets

En cliquant sur le menu « edit » (Modifier) d'un nœud d'application `MachineSet`, vous verrez que vous pouvez simplement provisionner une nouvelle machine virtuelle de nœud d'application en sélectionnant **Edit Machine Count** (Modifier le nombre de machines).

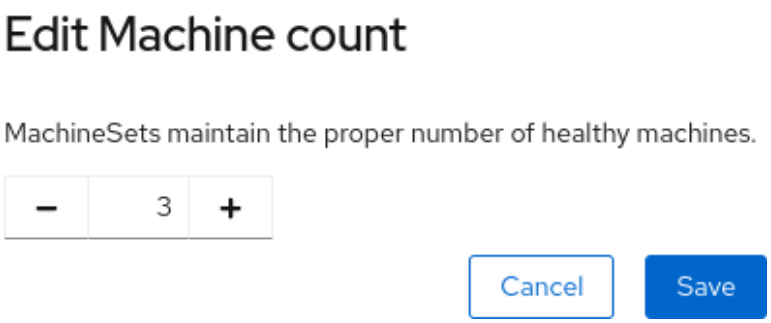


Figure 6.6 : Modification du nombre de machines

Une fois le nombre mis à jour, les administrateurs peuvent soit se rendre sur le portail Azure, soit sur la vue **Compute** → **Machines** pour afficher une nouvelle machine virtuelle de nœud d'application en cours de provisionnement.

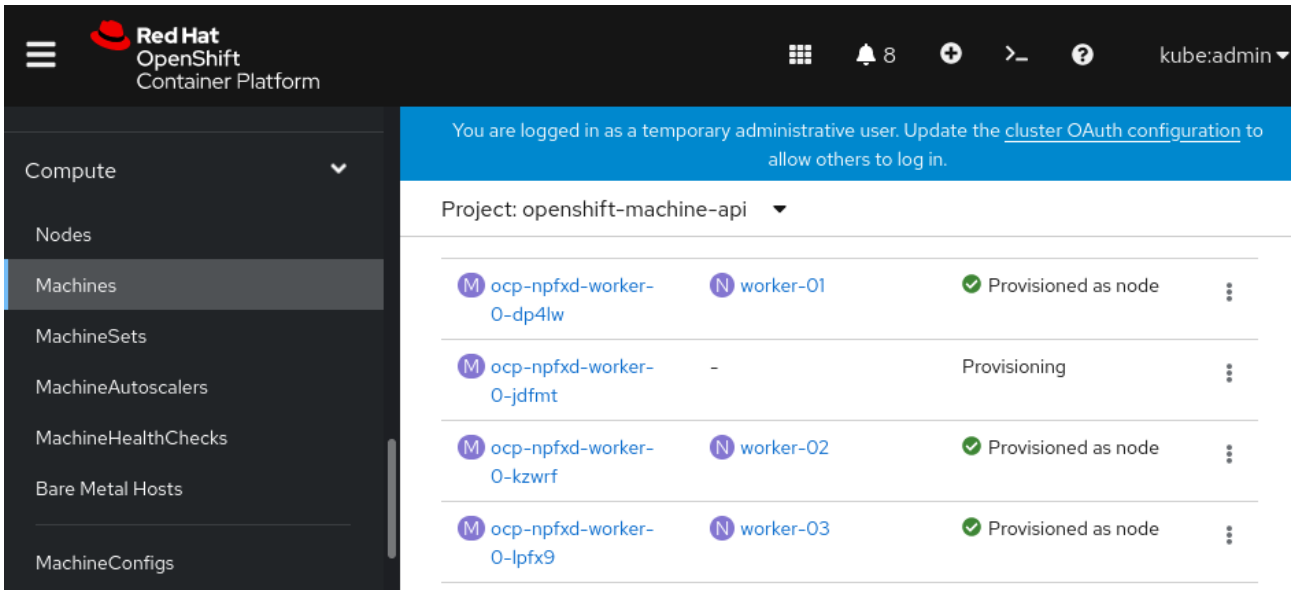


Figure 6.7 : Affichage des machines

En règle générale, le provisionnement d'une machine virtuelle supplémentaire prend jusqu'à cinq minutes dans la plupart des régions Azure.

Les administrateurs n'ont pas besoin d'effectuer d'activités de post-provisionnement pour mettre en ligne cette nouvelle capacité. OpenShift la mettra automatiquement à la disposition du cluster et les applications l'utiliseront en cas de besoin.

Mise à l'échelle automatique du cluster

La fonctionnalité de mise à l'échelle automatique n'est pas activée dans la configuration du déploiement par défaut d'Azure Red Hat OpenShift, mais elle est facile à activer. Les administrateurs doivent juste créer une ressource `MachineAutoscaler`, disponible dans le menu latéral **Compute** d'Azure Red Hat OpenShift.

Les ressources `MachineAutoscaler` fonctionnent sur un `MachineSet`. Ce dernier créera ou supprimera de la capacité des machines virtuelles des nœuds d'application selon les besoins. L'exemple suivant montre qu'il est possible d'assurer le bon fonctionnement d'un nombre minimum ou maximum de machines dans un `MachineSet` :

```
apiVersion: autoscaling.openshift.io/v1beta1
kind: MachineAutoscaler
metadata:
    name: worker-us-east-1a
    namespace: openshift-machine-api
spec:
    minReplicas: 1
    maxReplicas: 12
    scaleTargetRef:
        apiVersion: machine.openshift.io/v1beta1
        kind: MachineSet
        name: worker
```

OpenShift intègre également le concept d'évolutivité automatique des clusters qui permet la mise à l'échelle du système en fonction de la disponibilité d'une certaine quantité de RAM, de processeurs ou autres. Le choix entre `MachineAutoscaler` ou `ClusterAutoscaler` dépend de la manière dont vous souhaitez ajouter et supprimer de la capacité dans votre cluster.

Documentation de Red Hat OpenShift sur les MachineAutoscalers et ClusterAutoscalers

Mise à l'échelle d'applications

La mise à l'échelle des applications de microservices est un sujet complexe qui sort du cadre de ce manuel. Si vous souhaitez vous renseigner davantage à ce sujet, voici deux liens utiles :

- HorizontalPodAutoscaler : spécifiez le nombre minimum et maximum de pods que vous souhaitez exécuter, ainsi que l'utilisation du processeur ou de la mémoire de vos pods.
- Serverless et remise à zéro avec Knative.

Le cluster surveillera les conteneurs en cours d'exécution et la capacité restante sur le cluster. Si Knative ou `HorizontalPodAutoscaler` demande plus de ressources que celles actuellement disponibles sur le cluster, vous devrez procéder à la mise à l'échelle de `ClusterAutoscaler` ou de `MachineSet` pour ajouter les ressources demandées au cluster.

Grâce à cette approche, les applications et le cluster lui-même peuvent évoluer en tandem, à la hausse comme à la baisse, en fonction de la demande.

Configuration d'un secret d'extraction (enregistrement dans OpenShift Cluster Manager)

Lors d'un nouveau déploiement d'Azure Red Hat OpenShift, aucun secret d'extraction n'est configuré pour `cloud.redhat.com`. Cela signifie que vos clusters n'apparaîtront pas par défaut dans la console de cloud hybride Red Hat (http://console.redhat.com), c'est ce que nous appelons OpenShift Cluster Manager.

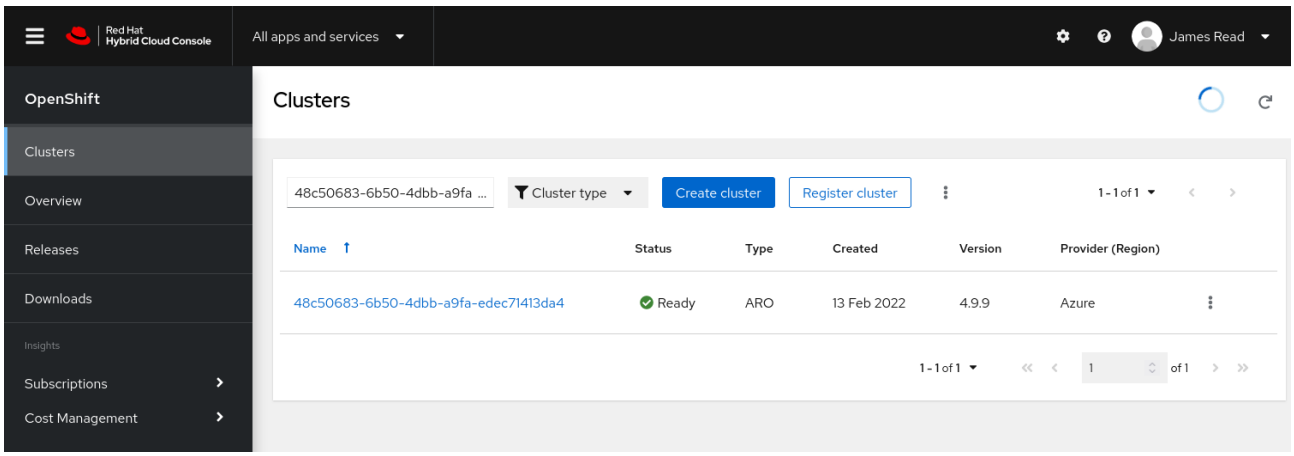


Figure 6.8 : OpenShift Cluster Manager présentant un cluster Azure Red Hat OpenShift

La configuration d'un secret d'extraction pour `cloud.redhat.com` est très simple. En plus d'être visible sur le portail OpenShift Cluster Manager à l'adresse http://console.redhat.com, vous pourrez également créer des tickets d'assistance directement avec Red Hat via le système de tickets d'assistance standard.

Vous trouverez les instructions pour la mise en place d'un secret d'extraction ici :

- Comment ajouter ou mettre à jour un secret d'extraction

La mise en place d'un secret d'extraction offre l'avantage supplémentaire de permettre aux clients de créer des tickets d'assistance directement auprès de Red Hat. Le secret d'extraction accorde un droit sur votre compte Red Hat qui permet au personnel d'assistance de voir votre cluster. L'ouverture d'un ticket d'assistance pour Azure Red Hat OpenShift est alors identique à celle de tout autre produit Red Hat.

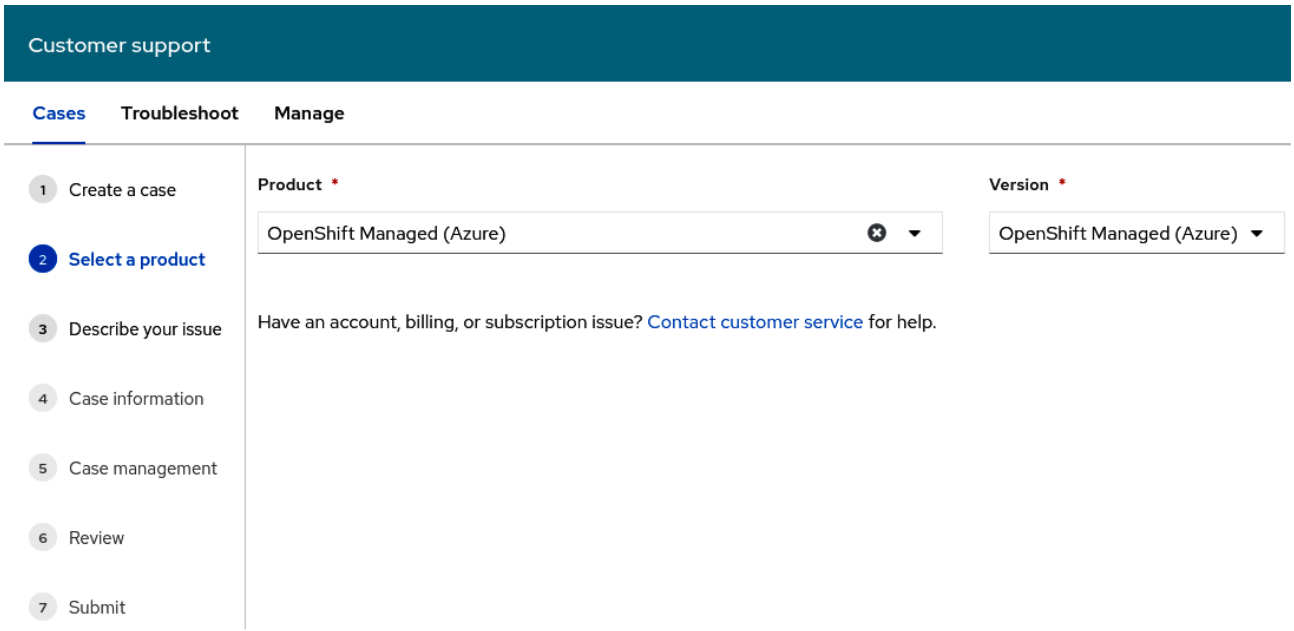


Figure 6.9 : Création de dossiers d'assistance

Plages de limites

Lorsqu'une équipe de développement ou d'application commence à déployer des applications conteneurisées, il s'écoule un certain temps avant qu'une application dysfonctionne et commence à consommer des ressources inutiles sur le cluster. Il peut s'agir, par exemple, d'une application défectueuse présentant une fuite de mémoire ou d'une application qui a été mise à l'échelle par erreur après avoir consommé seulement 10 % du processeur au lieu de 100 %. Afin d'éviter les scénarios où ces applications défaillantes évoluent de manière incontrôlée et consomment trop de ressources, nous vous recommandons d'utiliser `LimitRange`.

`LimitRange` permet de limiter la consommation de ressources d'objets spécifiques dans un projet. Applications de `LimitRange` :

- Pods et conteneurs : vous pouvez définir des exigences minimales et maximales en matière de processeur et de mémoire pour les pods et leurs conteneurs.
- Flux d'images : vous pouvez fixer des limites au nombre d'images et de balises dans un objet `ImageStream`.
- Images : vous pouvez limiter la taille des images qui peuvent être transmises à un registre interne.
- **Demandes de volume persistant (PVC) :** vous pouvez restreindre la taille des PVC qui peuvent être demandées.

Voici un exemple de plage de limites qui s'applique aux conteneurs, limitant les demandes de processeur et de mémoire minimales, maximales et par défaut autorisées :

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "resource-limits"
spec:
  limits:
    - type: "Container"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "100m"
        memory: "4Mi"
      default:
        cpu: "300m"
        memory: "200Mi"
      defaultRequest:
        cpu: "200m"
        memory: "100Mi"
      maxLimitRequestRatio:
        cpu: "10"
```

Ce snippet de code `LimitRange` peut être appliqué en copiant et en collant le YAML directement dans l'éditeur de la console Red Hat OpenShift ou à partir d'un fichier avec `oc apply -f limitrange.yaml`.

[Documentation sur les plages de limites]

Stockage persistant

Les machines virtuelles déployées par Azure Red Hat OpenShift intègrent des disques Azure pour installer Red Hat CoreOS et exécuter le service Azure Red Hat OpenShift. Seul le cluster OpenShift doit utiliser ces disques. Aucune application ne doit les utiliser.

Les applications qui nécessitent un stockage persistant doivent utiliser la fonction `PersistentVolume` de Kubernetes. La section sur la compréhension du stockage persistant de la documentation d'OpenShift décrit parfaitement bien ce concept en détail. Bien qu'Azure Red Hat OpenShift prenne techniquement en charge tous les fournisseurs `PersistentVolume` en tant qu'installation OpenShift autogérée, sur Azure, le stockage persistant le plus couramment utilisé est le suivant :

Nom	Type	Modes d'accès	Classe de stockage
Fichiers Azure	Système de fichiers, non compatible POSIX	ReadWriteOnce	Fichier Azure
Disque Azure	Bloc	ReadWriteOnce	Disque Azure
Red Hat OpenShift Data Foundation	Système de fichiers, bloc, objet	(plusieurs)	Docs OCS/ODF

Sécurité et conformité

La documentation de Red Hat OpenShift intègre désormais une quantité considérable de contenu pertinent pour comprendre comment mettre en œuvre de nombreux contrôles de sécurité tout en assurant la conformité.

Documentation sur la sécurité et la conformité d'OpenShift

Cette section de la documentation comprend :

- Une description détaillée du fonctionnement de la sécurité des conteneurs dans OpenShift. Vous devez comprendre qu'OpenShift offre de nombreux contrôles de sécurité prêts à l'emploi pour les conteneurs, que vous ne trouverez pas dans d'autres services basés sur Kubernetes, et que ces contrôles contribuent à garantir la sécurité de votre organisation et de vos applications.
- Recherche de vulnérabilités sur les pods
- Accès aux journaux d'audit
- Configuration des certificats

Cela inclut également plusieurs opérateurs utiles liés à la sécurité, tels que :

- **Opérateur de conformité :** exécutez des analyses et bénéficiez de recommandations pour remédier aux problèmes de sécurité courants.
- **Contrôle de l'intégrité des fichiers :** vérifiez en permanence que les fichiers, en particulier les fichiers de configuration de sécurité sensibles, n'ont pas été modifiés.

Synthèse

Ce chapitre a couvert la plupart des tâches et des thèmes dont les organisations tiennent généralement compte pour adapter aux applications de production un cluster nouvellement déployé. Bien qu'il ne soit pas nécessaire de passer en revue chacun de ces thèmes à chaque déploiement ultérieur, le premier cluster que vous déployez doit prendre en compte le stockage persistant, les limites, l'authentification et les divers autres thèmes mentionnés dans ce chapitre.

En général, une organisation essaiera d'automatiser les tâches de post-provisionnement autant que possible. Par exemple, vous pouvez généralement installer et configurer Azure Active Directory à l'aide d'un script PowerShell ou de quelques modèles ARM. Vous pouvez ainsi réduire le temps consacré aux tâches répétitives si vous déployez de nombreux clusters.

Le chapitre suivant de ce manuel présente un exemple de déploiement d'une application sur votre cluster Azure Red Hat OpenShift prêt pour la production.

Chapitre 7

Déploiement d'une application (exemple)

Le contenu de ce manuel est principalement destiné à aider le public technique (développeurs et responsables des opérations) qui cherche à comprendre ce dont il a besoin pour utiliser Azure Red Hat OpenShift en production. Ce manuel suppose que le lecteur a une connaissance de base de Red Hat OpenShift, car une grande partie de l'architecture, des portails de gestion et de l'expérience utilisateur est identique à Azure Red Hat OpenShift.

Ce chapitre est une introduction rapide qui explique comment déployer une application simple à partir d'un exemple : l'application « Fruit Smoothies ». Cette application sert de démarrage rapide et de rappel pour vous permettre de mieux comprendre comment utiliser Azure Red Hat OpenShift. Le déploiement de cette application, compatible avec Azure Kubernetes Service, sur Azure Red Hat OpenShift est la preuve qu'OpenShift est totalement compatible avec Kubernetes.

Ce chapitre s'appuie largement sur le contenu de http://aroworkshop.io. Des descriptions et un contexte supplémentaires ont été ajoutés.

Un aperçu de l'application d'évaluation

L'architecture de l'application est simple :

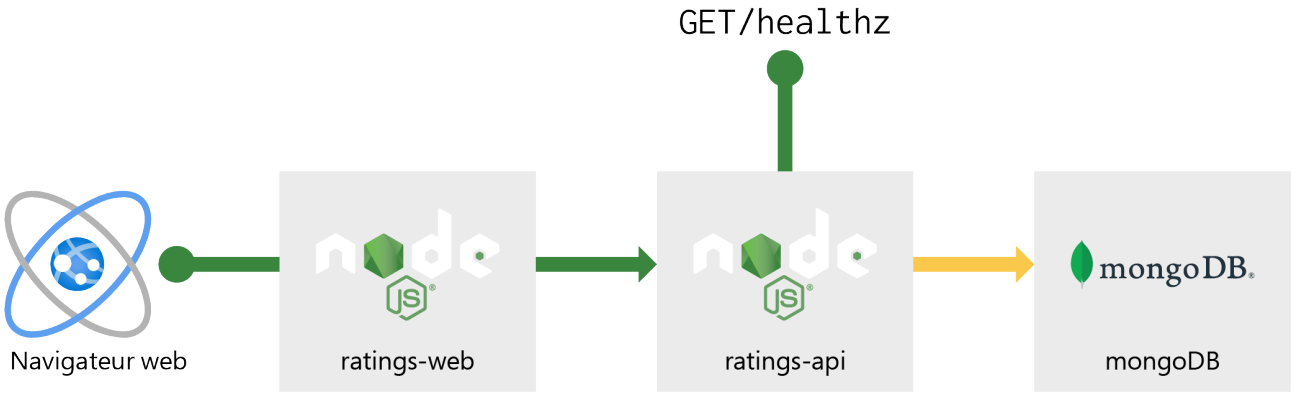


Figure 7.1 : Architecture de l'application Fruit Smoothies

Le schéma précédent montre que l'application se compose de trois services et de deux points de terminaison publics, comme indiqué dans le tableau ci-dessous. Le premier point de terminaison sur la gauche du schéma représente le navigateur web d'un utilisateur et montre l'application web HTML publique. Le second point de terminaison (`healthz`) est un contrôle d'intégrité du service `ratings-api`. Le tableau décrit les différents services et fournit les liens vers leurs référentiels :

Composant	Description	Référentiel GitHub
`rating-web`	Un front-end web public : le « site web ».	Réf. GitHub
`rating-api`	Ce service prend les données de l'interface utilisateur web et les stocke dans la base de données. Il renvoie également les résultats de la base de données à l'application web sur le port 3000.	Réf. GitHub
`mongodb`	Une base de données NoSQL avec des données préchargées.	Données

Vous pouvez consulter les liens du référentiel GitHub pour étudier et mieux comprendre ces applications. Les instructions suivantes proposent des conseils, étape par étape, sur la manière de déployer chacune de ces applications.

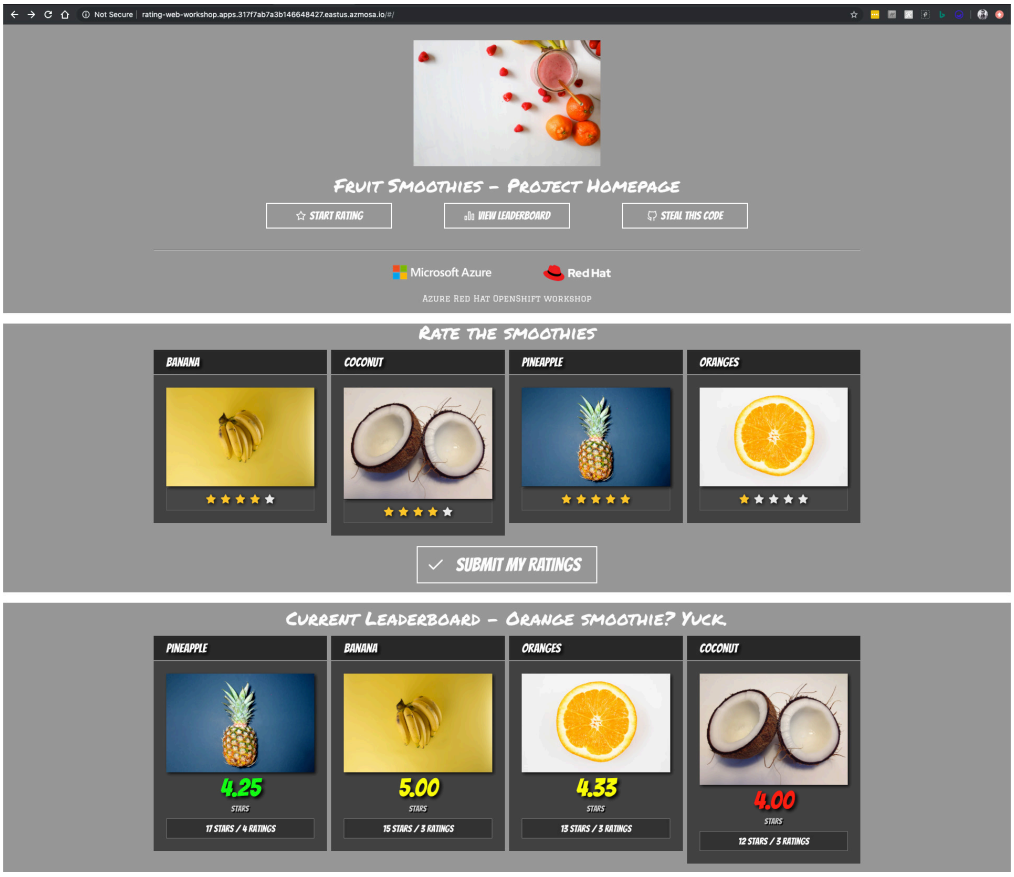


Figure 7.2 : Captures d'écran montrant un aperçu de l'application « Fruit Smoothies »

Une fois que vous aurez terminé, votre application web ressemblera aux captures d'écran précédentes. Vous devriez également mieux comprendre comment les développeurs et les équipes d'exploitation déploient des applications sur Azure Red Hat OpenShift. Cela devrait faciliter votre propre processus décisionnel lorsque vous déployez vous-même des applications sur Azure Red Hat OpenShift.

Créer le cluster et s'y connecter

Le reste de ce chapitre suppose que vous disposez d'un environnement Azure Red Hat OpenShift fonctionnel pour travailler. Cette application d'évaluation n'implique aucune exigence particulière. Son déploiement se fait dans un environnement Azure Red Hat OpenShift vierge et dynamique. Si vous n'avez pas encore provisionné de cluster, consultez les chapitres suivants :

- *Chapitre 4 : Pré-provisionnement – Questions sur l'architecture d'entreprise*
- *Chapitre 5 : Provisionnement d'un cluster Azure Red Hat OpenShift*

La section *Accès au cluster* du *Chapitre 5 : Provisionnement d'un cluster Azure Red Hat OpenShift* offre un rappel utile de la procédure d'accès au cluster que vous avez provisionné précédemment.

Seconnecter à la console web

Chaque cluster Azure Red Hat OpenShift possède une adresse DNS pour la console web OpenShift. Vous pouvez utiliser la commande `az aro list` pour répertorier les clusters de votre abonnement Azure actuel :

```
az aro list -o table
```

L'URL de la console web du cluster sera ajoutée à la liste. Ouvrez ce lien dans un nouvel onglet du navigateur et connectez-vous avec l'utilisateur `kubeadmin` ou un autre compte utilisateur autorisé à créer des projets.

Une fois connecté, la console web Azure Red Hat OpenShift devrait s'afficher.

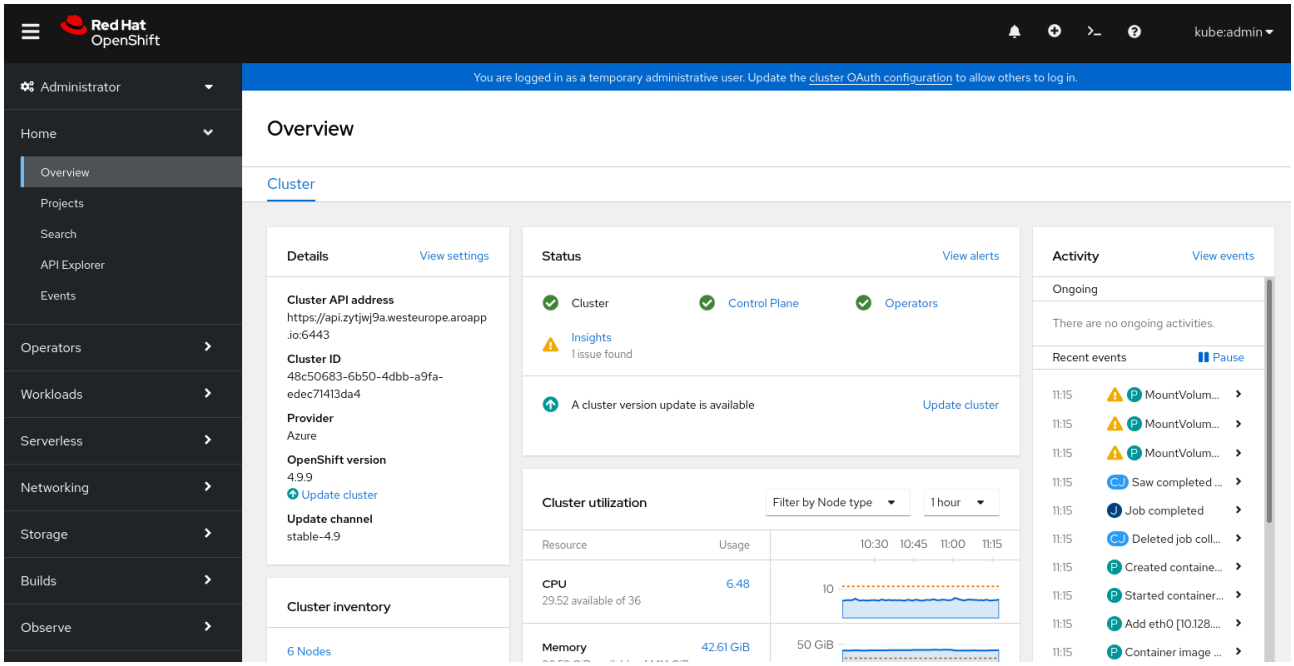


Figure 7.3 : Console web Azure Red Hat OpenShift

Installer le client OpenShift

Ouvrez [Azure Cloud Shell] ou utilisez votre terminal Linux local et installez le client de ligne de commande OpenShift. Cette action est obligatoire pour accéder au cluster depuis la ligne de commande. Voici les instructions requises :

```
cd ~
curl https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-linux.tar.gz > openshift-client-linux.tar.gz

mkdir openshift

tar -zxvf openshift-client-linux.tar.gz -C openshift

echo 'export PATH=$PATH:~/openshift' >> ~/.bashrc && source ~/.bashrc
```

Sinon, voici les liens de téléchargement pour Windows ou Mac :

- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-windows.zip
- https://mirror.openshift.com/pub/openshift-v4/clients/ocp/latest/openshift-client-mac.tar.gz

Vous devriez être en mesure d'exécuter la commande oc à partir de l'un de ces paquets téléchargés.

Récupérer la commande et le jeton de connexion

Après l'installation du client, il est nécessaire d'obtenir un jeton pour se connecter au cluster. Connectez-vous à la console web OpenShift, cliquez sur le nom d'utilisateur en haut à droite, puis cliquez sur **Copy Login Command** (Copier la commande de connexion).

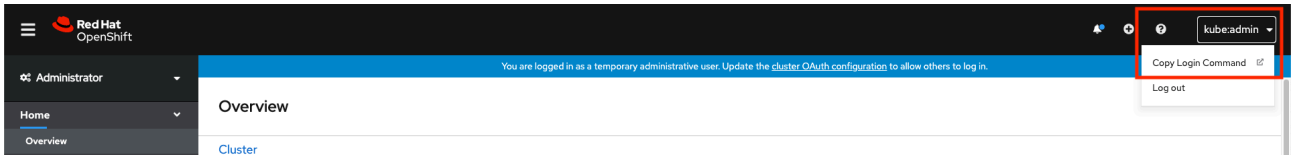


Figure 7.4 : Copier la commande de connexion pour se connecter au cluster

Collez la commande de connexion dans votre shell (terminal Linux local ou Azure Cloud Shell). Vous devriez pouvoir vous connecter au cluster.

Figure 7.5 : Utiliser la commande de connexion pour se connecter au cluster

Une fois connecté, passez à la création d'un projet.

Créer un projet

Un projet dans OpenShift ressemble à un dossier logique pour contenir l'application d'évaluation. Dans Red Hat OpenShift, tous les conteneurs et applications doivent se trouver dans un projet quelconque. Vous pouvez utiliser les projets pour séparer les applications, voire les services. Les quelques instructions suivantes expliquent comment créer un projet.

Bien que vous puissiez créer un projet à partir de l'interface web, ces instructions utilisent la ligne de commande :

```
oc new-project workshop
```

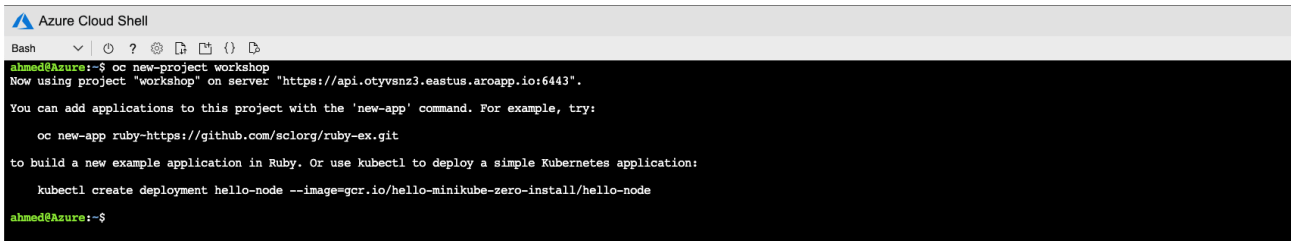


Figure 7.6 : Créer un nouvel atelier dans Azure Cloud Shell

Une fois le projet créé, vous pouvez basculer dessus avec `oc project workshop`. L'étape suivante consistera à déployer le premier des trois microservices présentés au début de ce chapitre : MongoDB. Vous le déploierez dans le projet que nous venons de créer.

Ressources

- Documentation d'Azure Red Hat OpenShift – Mise en route avec l'interface en ligne de commande
- Documentation d'Azure Red Hat OpenShift – Projets

Déployer MongoDB

Azure Red Hat OpenShift fournit une image de conteneur et un modèle pour faciliter la création d'un nouveau service de base de données MongoDB. Le modèle fournit des champs de paramètres pour définir toutes les variables d'environnement obligatoires (utilisateur, mot de passe, nom de la base de données, etc.) avec des valeurs par défaut prédéfinies, y compris l'autogénération des valeurs de mot de passe. Il définit également une configuration de déploiement et un service.

L'instance MongoDB sera déployée directement en tant qu'image de conteneur à partir de Docker Hub. Grâce à OpenShift, vous pouvez procéder entièrement via la console web. Basculez vers la vue développeur en haut du menu, accédez à la page **Add** (Ajouter) et sélectionnez **Container images** (Images du conteneur).

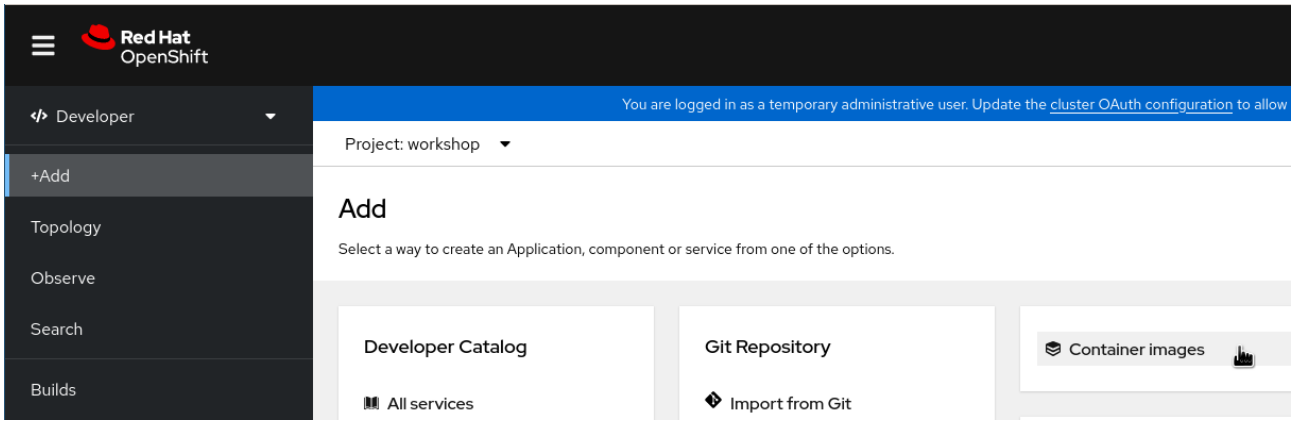


Figure 7.7 : Ajout d'une image de conteneur

La page **Deploy Image** (Déployer l'image) s'affiche. Remplissez le formulaire comme suit :

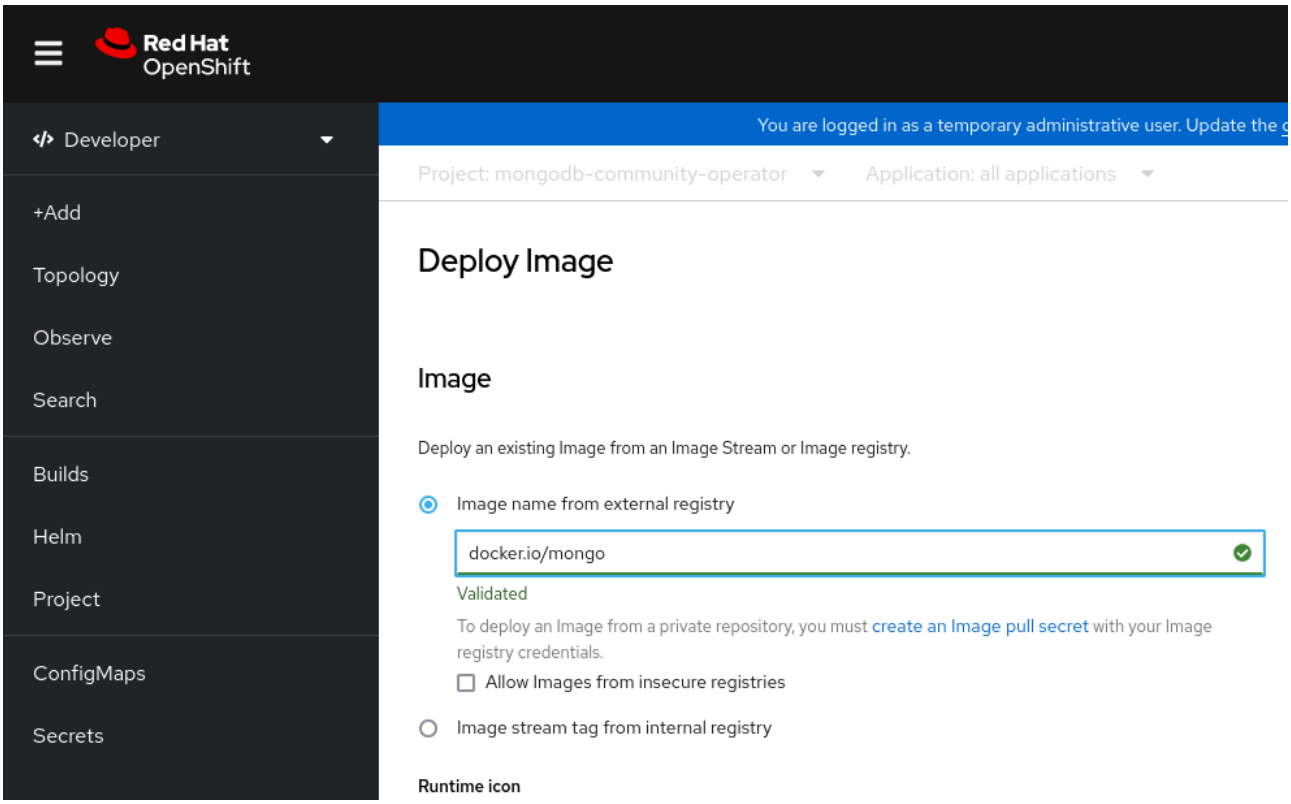


Figure 7.8 : Remplir le formulaire pour déployer une image

Veillez à définir les valeurs du formulaire comme suit :

Champ	Valeur
Image name from external registry (Nom de l'image à partir du registre externe)	`docker.io/mongo`
Icône Runtime (Exécution)	`mongodb`
Application name (Nom de l'application)	`ratings`
Name (Nom)	`mongodb`
Create route to application (Créer un routage vers l'application)	Option non cochée. Un routage permettrait d'accéder à la base de données en externe, ce qui n'est pas nécessaire pour cette application
Resource type (Type de ressource)	Déploiement

Lorsque vous arrivez au bas du formulaire, sélectionnez le lien de déploiement pour développer le formulaire et définir ainsi des variables d'environnement.

La variable d'environnement suivante agit par défaut pour initialiser la base de données lors de son premier démarrage :

```
Name                    Value
MONGO_INITDB_DATABASE   ratingsdb
```

Aucun nom d'utilisateur ni mot de passe n'est défini pour la base de données MongoDB. Il s'agit de la configuration par défaut et l'authentification sera désactivée.

Vérifiez à nouveau la variable d'environnement, puis cliquez sur le bouton **Create** (Créer) pour continuer et déployer le conteneur MongoDB.

Après quelques instants, l'instance MongoDB devrait être opérationnelle dans le projet `workspace`. Vous pouvez visualiser ce déploiement en basculant vers la vue **Topology** (Topologie).

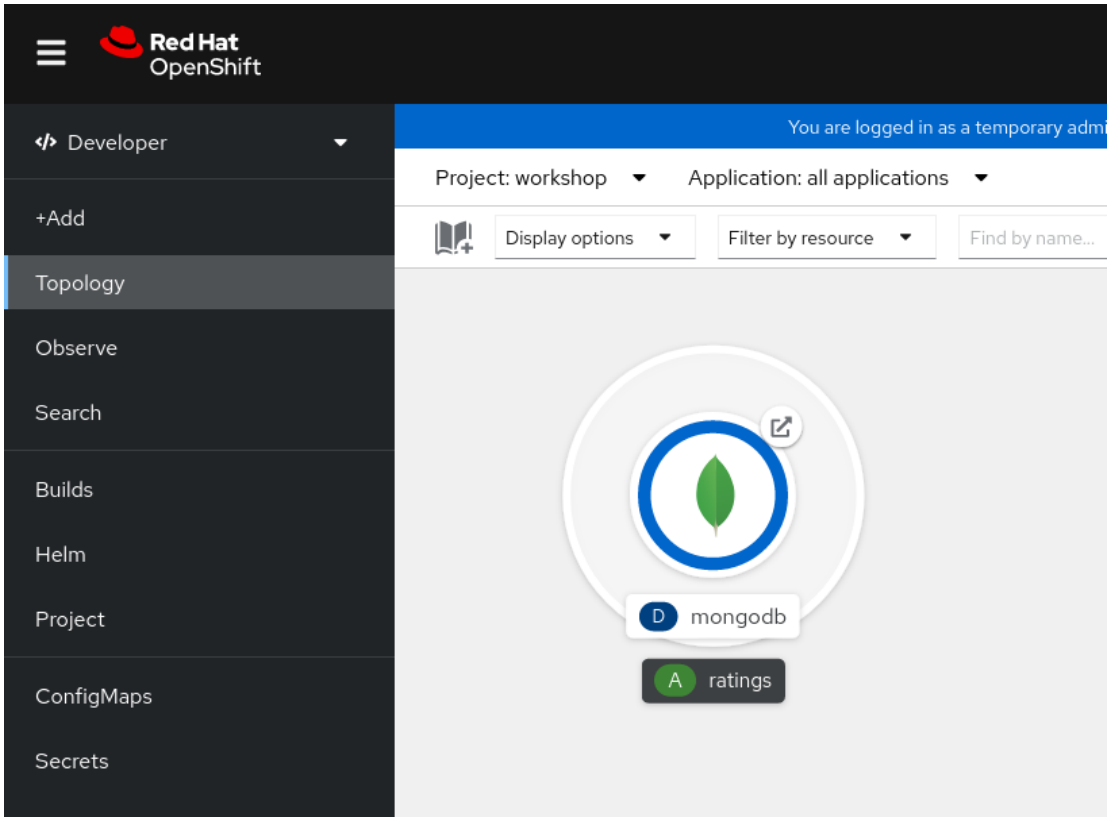


Figure 7.9 : Instance MongoDB opérationnelle

Exécutez la commande `oc get all` pour afficher l'état de la nouvelle application et vérifier si le déploiement du modèle MongoDB s'est bien déroulé. Exemple de résultat de la commande `oc get all` :

```
user@host: oc get all
NAME                              READY   STATUS    RESTARTS   AGE
pod/mongo-6c6fcb45b8-8wpdm        1/1     Running   0          29s

NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
service/mongo     ClusterIP   172.30.88.119    <none>        27017/TCP   30s

NAME                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mongo       1/1     1            1           30s

NAME                                    DESIRED   CURRENT   READY   AGE
replicaset.apps/mongo-6c6fcb45b8        1         1         1       30s

NAME                                        IMAGE REPOSITORY
TAGS      UPDATED
imagestream.image.openshift.io/mongo    image-registry.openshift-image-registry.svc:5000/workshop/mongo
latest    30 seconds ago
```

Si tout fonctionne correctement, la colonne STATUS (ÉTAT) devrait indiquer que le conteneur est en cours d'exécution.

Récupérer le nom d'hôte du service MongoDB

Une fois le déploiement terminé, nous devons rechercher le service qui a été créé pour nous permettre d'accéder à la base de données depuis le cluster. `svc` est l'abréviation de `services` :

```
user@host: oc get svc mongodb
NAME     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)     AGE
mongo    ClusterIP   172.30.88.119    <none>        27017/TCP   77s
```

Le service sera accessible sous le nom DNS suivant, `mongodb.workshop.svc.cluster.local`, qui est formé de `[nom du service].[nom du projet].svc.cluster.local`. La résolution n'est viable qu'au sein du cluster.

Déployer l'API d'évaluation

Vous devez maintenant déployer la deuxième application : `rating-api`. Il s'agit d'une application Node.js qui se connecte à une instance MongoDB pour récupérer et évaluer les éléments. Voici quelques-uns des détails dont vous aurez besoin pour déployer cette application :

- `rating-api` sur GitHub
- Le conteneur expose le port `3000`
- Une connexion MongoDB est configurée à l'aide d'une variable d'environnement appelée `MONGODB_URI`

Notez ces détails, car vous devrez vous y référer dans les sections suivantes.

Forker l'application dans votre dépôt GitHub

Pour effectuer des modifications, comme l'ajout de webhooks CI/CD, vous aurez besoin de votre propre copie du code `ratings-api`. Dans Git, cette copie s'appelle un « fork ». Vous pouvez forker l'application dans votre dépôt GitHub personnel. Accédez au dépôt GitHub précédent et cliquez sur le bouton **Fork** (Forker).

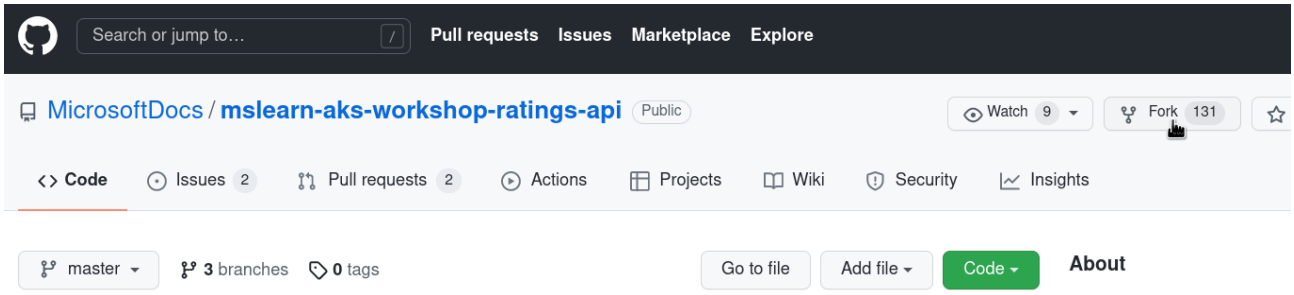


Figure 7.10 : Forker l'application dans le dépôt GitHub

Notez la nouvelle adresse du dépôt : vous devrez l'utiliser dans les instructions ci-après.

Utiliser l'interface en ligne de commande OpenShift pour déployer rating-api

OpenShift peut déployer du code directement à partir d'un dépôt Git en examinant le contenu et en sélectionnant une « image d'outil de création » en fonction de ce dernier : Java, PHP, Perl, Python ou autre. Ici, `rating-api` est une application JavaScript. L'image d'outil de création téléchargera les dépendances JavaScript avec npm et créera une nouvelle image de conteneur stockée dans le registre interne des conteneurs d'OpenShift. Cette stratégie de build s'appelle **Source 2 Image (S2I)**. Sa description détaillée figure dans le glossaire.

Vous pouvez lancer une nouvelle build S2I avec `oc new-app` :

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-api
--strategy=source --name=rating-api

--> Found image 0aea15f (3 weeks old) in image stream "openshift/nodejs" under tag "14-ubi8" for
"nodejs"

    Node.js 14
    ----------
     Node.js 14 available as container is a base platform for building and running various Node.
js 14 applications and frameworks. Node.js is a platform built on Chrome's JavaScript runtime
for easily building fast, scalable network applications. Node.js uses an event-driven, non-
blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time
applications that run across distributed devices.

    Tags: builder, nodejs, nodejs14

    * The source repository appears to match: nodejs
    * A source build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-api will be created
      * The resulting image will be pushed to image stream tag "rating-api:latest"
      * Use 'oc start-build' to trigger a new build
```

Basculez vers la vue **Topology** (Topologie) dans la console web. Vous devriez voir la build d'application démarrer et l'exécution réussie du déploiement après quelques minutes.

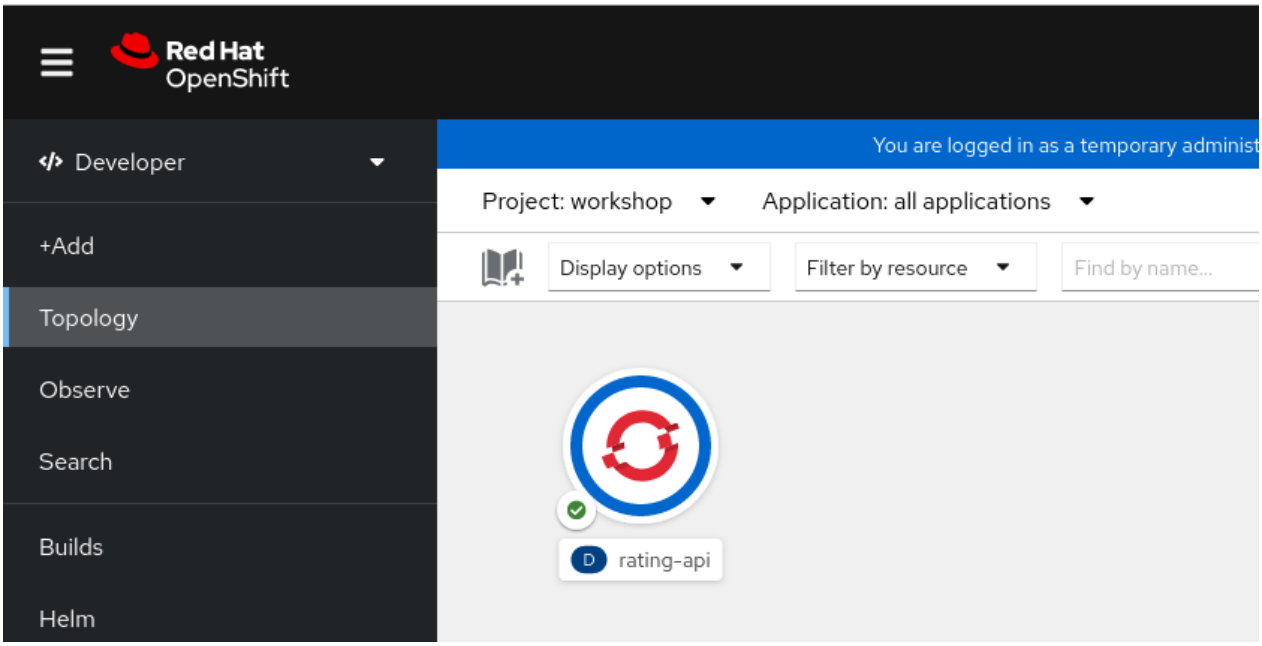


Figure 7.11 : Vue Topology (Topologie)

La build et le démarrage du pod ne devraient prendre qu'une ou deux minutes.

Configurer les variables d'environnement requises

À ce stade, la base de données est déployée, de même que l'API d'évaluation. Vous devez toutefois indiquer à l'API d'évaluation comment se connecter à la base de données. La configuration des applications basées sur des conteneurs est généralement réalisée à l'aide de variables d'environnement.

Cliquez sur le déploiement et modifiez-le. Créez la variable d'environnement suivante :

```
Name          Value
MONGODB_URI   mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

L'enregistrement de cette nouvelle variable d'environnement déclenchera un redéploiement du service `ratings-api` pour l'utiliser.

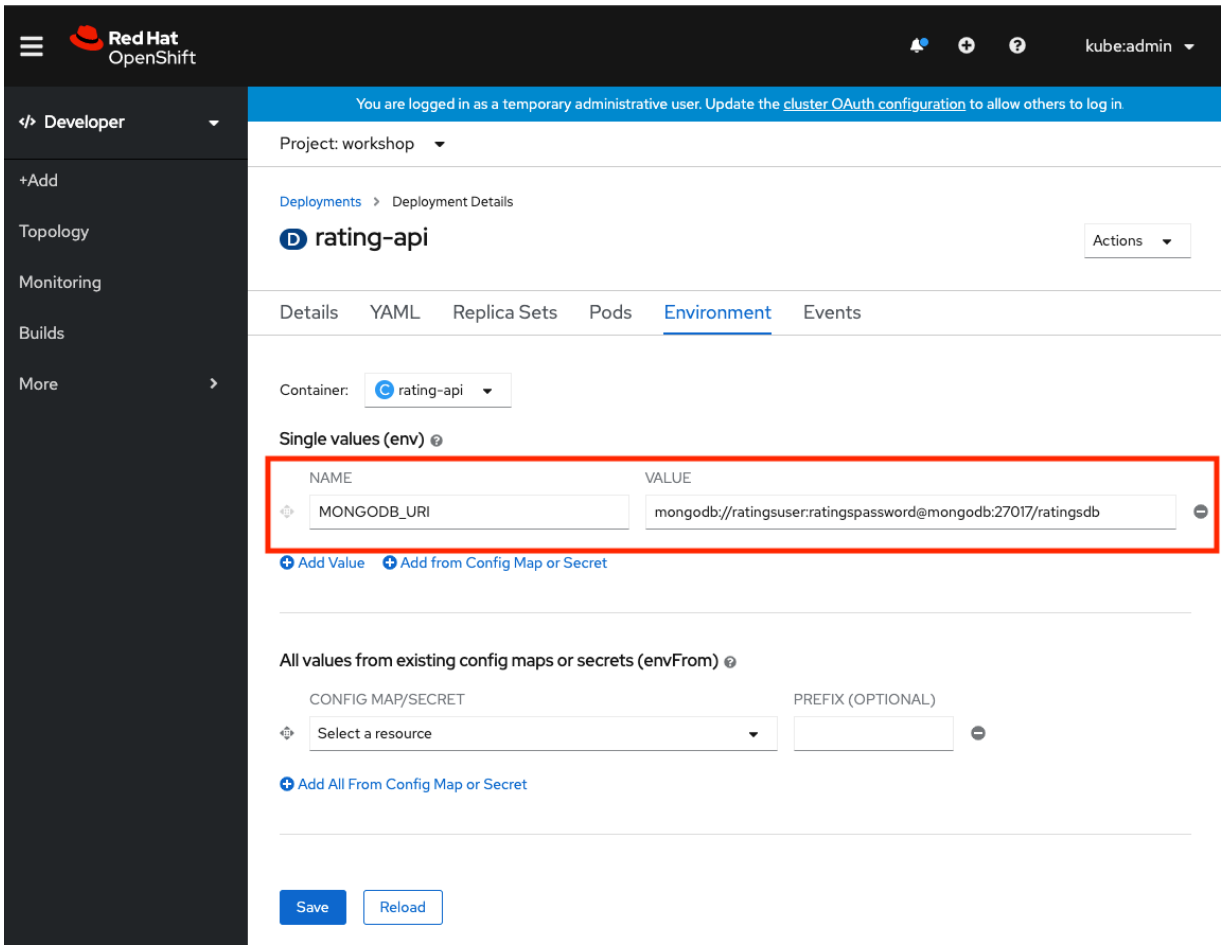


Figure 7.12 : Définition de la variable d'environnement MONGODB_URI via la console web

Pour ce faire, vous pouvez également utiliser la ligne de commande :

```
oc set env deploy/rating-api MONGODB_URI=mongodb://mongodb.workshop.svc.cluster.local:27017/ratingsdb
```

Quelle que soit la méthode utilisée, OpenShift devra redémarrer le conteneur pour utiliser les nouvelles variables d'environnement.

Vérifier que le service fonctionne

Si vous accédez aux journaux du déploiement de `rating-api`, vous devriez voir un message de journal confirmant que le code peut se connecter avec succès à MongoDB. Pour ce faire, dans l'écran de détails du déploiement, cliquez sur l'onglet **Pods**, puis sur l'un des pods.

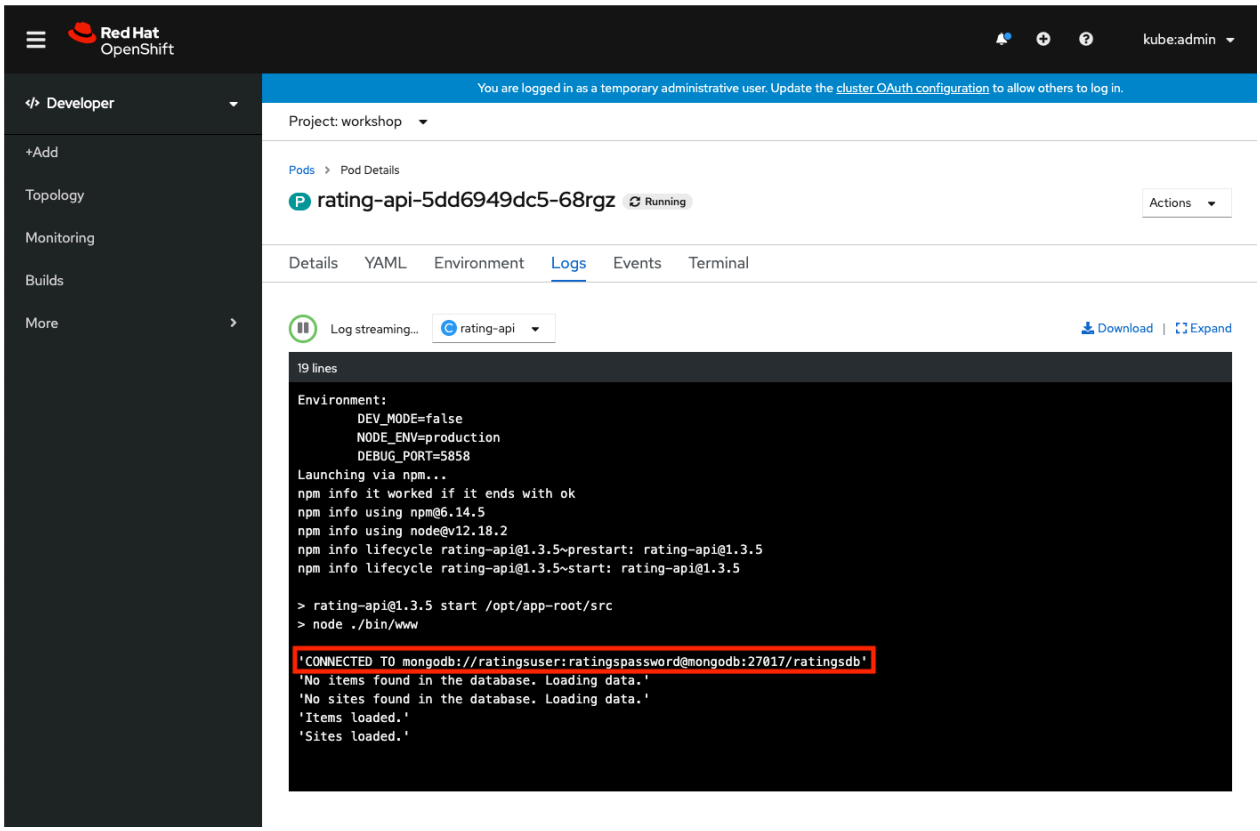


Figure 7.13 : Message du journal confirmant la connexion réussie du code à MongoDB

Ajuster le port de service rating-api

OpenShift créera un service utilisant le port `8080`. Cependant, suite à une mise à jour de la bibliothèque, ce service utilise le port `3000`. Vous devez modifier le service par défaut.

Accédez au menu **Networking** (Mise en réseau) → **Services**. Dans la liste des services, modifiez le service `rating-api` comme suit (remplacez simplement `8080` par `3000`) :

```
  ports:
    - name: 3000-tcp
      protocol: TCP
      port: 3000
      targetPort: 3000
```

Redémarrez le service pour utiliser le nouveau port :

```
user@host: oc rollout restart deploy/rating-api
```

Le service est désormais lié au port approprié : `3000` et non `8080`.

Récupérer le nom d'hôte du service rating-api

Nous devrons valider la présence d'un service `rating-api`. Ce dernier sera utilisé dans la section suivante, lors du déploiement de l'application `rating-web` :

```
oc get service rating-api
```

Le service sera accessible sous le nom DNS `rating-api.workshop.svc.cluster.local:3000`, sur le port `3000`, qui est formé de `[nom du service].[nom du projet].svc.cluster.local`. La résolution n'est viable qu'au sein du cluster.

Déployer le front-end d'évaluation

`rating-web` est une application Node.js qui se connecte à `rating-api`. Voici quelques-uns des détails dont vous aurez besoin pour déployer cette application :

- `rating-web` sur GitHub
- Le conteneur expose le port `8080`
- L'application web se connecte à l'API via le DNS interne du cluster, à l'aide d'un proxy au moyen d'une variable d'environnement nommée `API`.

Utiliser l'interface en ligne de commande OpenShift pour déployer rating-web

Comme pour l'application `rating-api`, vous pouvez déployer cette application avec S2I avec`oc new-app`. Cependant, cette fois-ci, l'application sera créée avec une stratégie Dockerfile avec un fichier Dockerfile qui figure déjà dans le dépôt Git. Par conséquent, ne spécifiez pas l'argument `--strategy` :

```
user@host: oc new-app https://github.com/<your GitHub username>/mslearn-aks-workshop-ratings-web
--name rating-web

--> Found container image e1495e4 (2 years old) from Docker Hub for "node:13.5-alpine"

    * An image stream tag will be created as "node:13.5-alpine" that will track the source image
    * A Docker build using source code from https://github.com/MicrosoftDocs/mslearn-aks-
workshop-ratings-web will be created
      * The resulting image will be pushed to image stream tag "rating-web:latest"
      * Every time "node:13.5-alpine" changes a new build will be triggered

--> Creating resources ...
    imagestream.image.openshift.io "node" created
    imagestream.image.openshift.io "rating-web" created
    buildconfig.build.openshift.io "rating-web" created
    deployment.apps "rating-web" created
    service "rating-web" created
--> Success
```

La création des dépendances dans un conteneur prendra un certain temps. Comptez environ 2 à 3 minutes avant qu'elle ne se termine et que la vue **Topology** (Topologie) s'affiche, montrant le service en ligne.

Configurer les variables d'environnement requises

Créez la variable d'environnement `API` pour la configuration de déploiement de `rating-web`. La valeur de cette variable sera le nom d'hôte et le port du service `rating-api`.

Au lieu de définir la variable d'environnement par le biais de la console web Azure Red Hat OpenShift, vous pouvez la définir par le biais de l'interface en ligne de commande OpenShift :

```
oc set env deploy rating-web API=http://rating-api:3000
```

Exposer le service rating-web avec un routage

Exposer le service signifie qu'une URL publique existante permet aux utilisateurs d'accéder au service. Si le service n'est pas exposé, il n'est accessible qu'au sein du cluster :

```
user@host: oc expose svc/rating-web
route.route.openshift.io/rating-web exposed
```

Enfin, obtenez l'URL du service que vous venez d'exposer :

```
user@host: oc get route rating-web
NAME          HOST/PORT                                                PATH    SERVICES      PORT
TERMINATION   WILDCARD
rating-web    rating-web-workshop.apps.zytjwj9a.westeurope.aroapp.io          rating-web    8080-tcp
None
```

Notez que le **nom de domaine complet (FQDN)** inclut le nom de l'application et le nom du projet par défaut. Le reste du FQDN, soit le sous-domaine, correspond à votre sous-domaine d'applications spécifique au cluster Azure Red Hat OpenShift.

Essayer le service

Ouvrez le nom d'hôte dans votre navigateur. Vous devriez voir la page de l'application d'évaluation. Testez, soumettez quelques votes et consultez le classement.

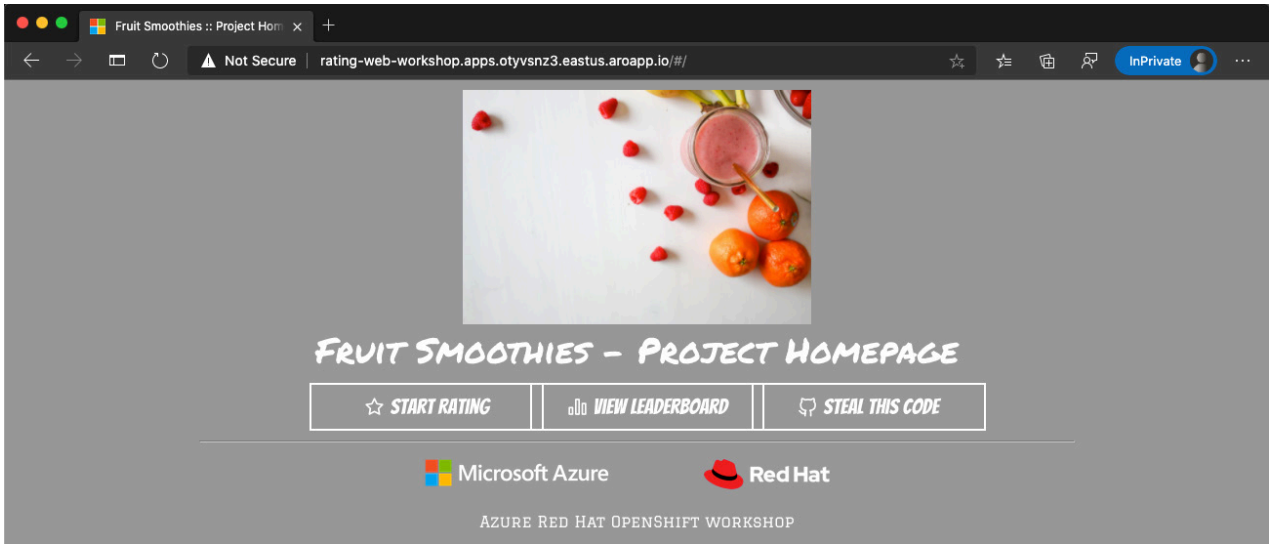


Figure 7.14 : Test du service d'application d'évaluation

Configurer le webhook GitHub

Pour déclencher les builds de S2I lorsque vous poussez du code dans votre dépôt GitHub, vous devez configurer le webhook GitHub :

1. Récupérez le secret de déclenchement du webhook de GitHub. Vous devrez utiliser ce secret dans l'URL du webhook GitHub :

```
user@host: oc get bc/rating-web -o=jsonpath='{.spec.triggers..github.secret}'
3ffcc8d5-a243
```

   Notez la clé secrète, car vous en aurez besoin dans quelques étapes.

2. Récupérez l'URL du déclencheur webhook de GitHub à partir de la configuration de la build :

```
user@host: oc describe bc/rating-web
...
Webhook GitHub:
        URL:    https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/
namespaces/workshop/buildconfigs/rating-web/webhooks/<secret>/github
...
```

3. Remplacez l'espace réservé `<secret>` par le secret que vous avez récupéré à la première étape. Ici, le secret est `3ffcc8d5-a243` et l'URL qui en résulte est :

```
https://api.quwhfg7o.westeurope.aroapp.io:6443/apis/build.openshift.io/v1/namespaces/workshop/
buildconfigs/rating-web/webhooks/3ffcc8d5-a243/github
```

   Vous utiliserez cette URL pour configurer le webhook sur votre dépôt GitHub.

4. Accédez à votre dépôt GitHub. Sélectionnez **Add Webhook** (Ajouter webhook) sous **Settings** (Paramètres) → **Webhooks**.
5. Dans le champ **Payload URL** (URL de charge utile), collez votre URL GitHub avec la valeur `<secret>` modifiée pour utiliser votre secret.
6. Dans le champ **Content type** (Type de contenu), remplacez la valeur `application/x-www-form-urlencoded` par défaut par `application/json`.

7. Cliquez sur **Add webhook** (Ajouter webhook).

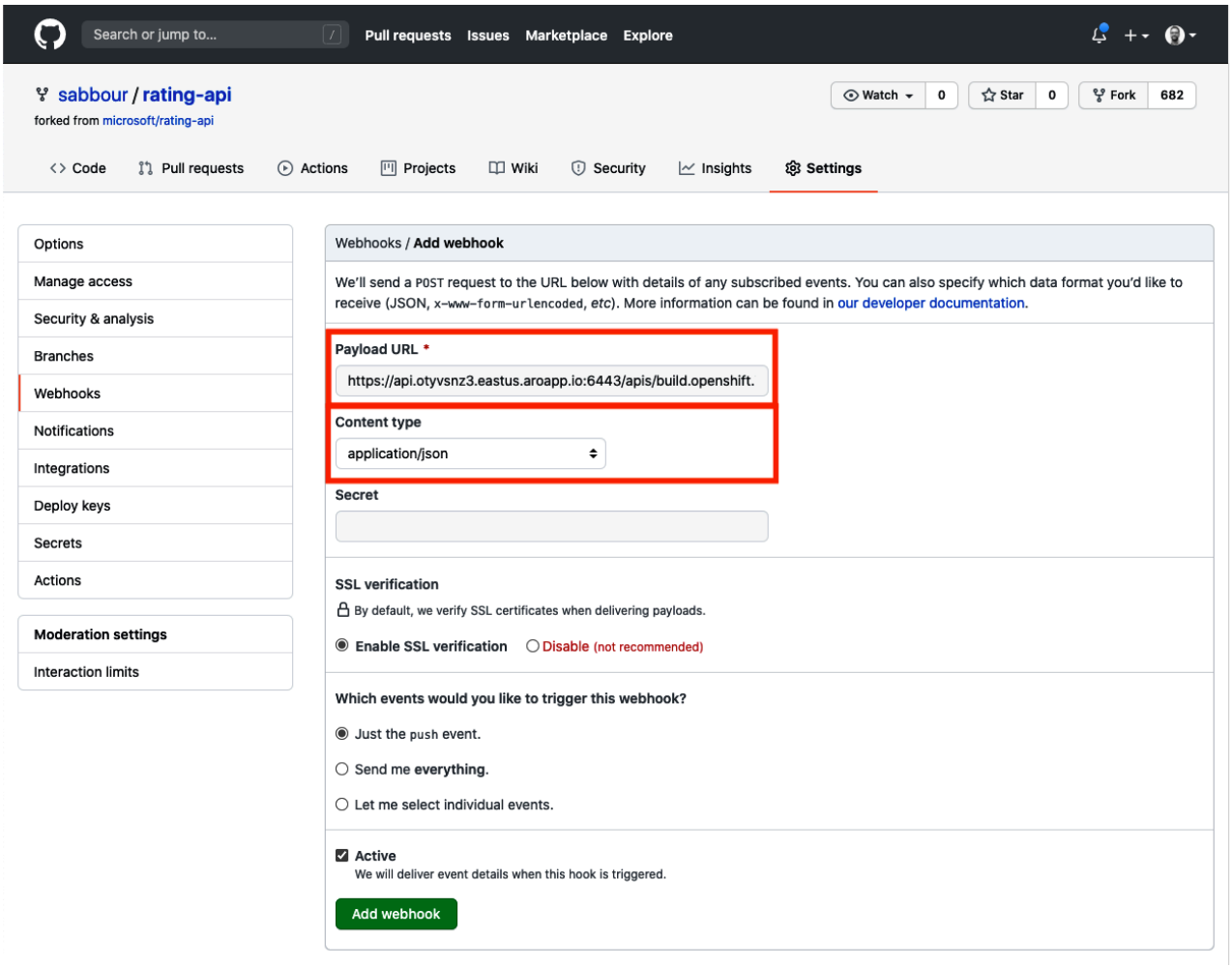


Figure 7.15 : Ajouter un webhook

Un message de GitHub qui indique que votre webhook a été configuré avec succès doit s'afficher.

Désormais, chaque modification apportée à votre dépôt GitHub lancera automatiquement une nouvelle build. Une fois la build lancée, un nouveau déploiement commencera.

Modifier l'application du site web et voir la mise à jour continue

Accédez au fichier `https://github.com/<your GitHub username>/rating-web/blob/master/src/App.vue` dans votre dépôt sur GitHub.

Modifiez le fichier ; remplacez la ligne `background-color: #999;` par `background-color: #0071c5`.

Validez les modifications du fichier dans la branche `master`.

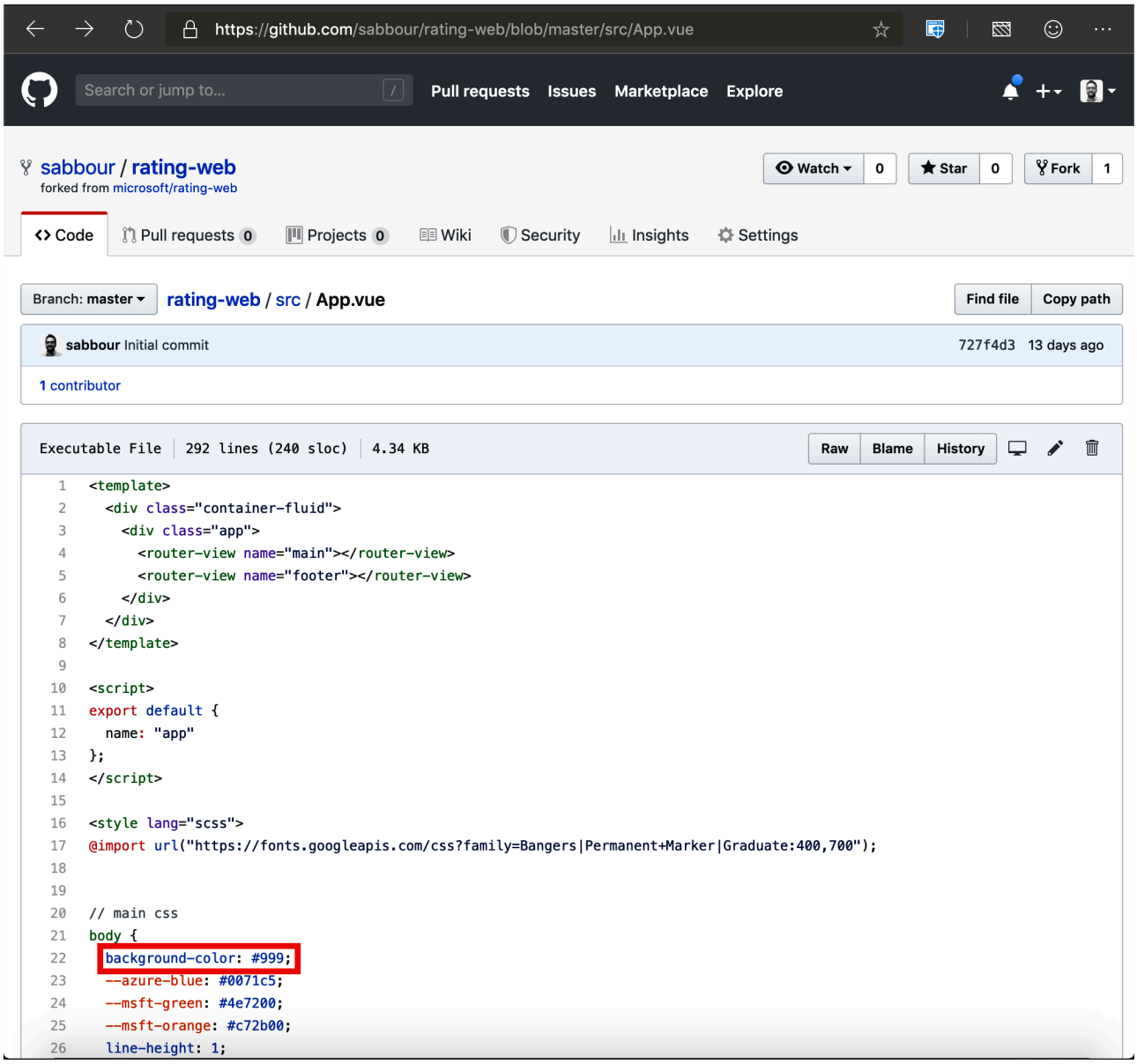


Figure 7.16 : Validation des modifications du fichier dans la branche

Accédez immédiatement à l'onglet **Builds** de la console web OpenShift. Une nouvelle build est visible dans la file d'attente. Elle a été déclenchée par cette validation. Son exécution déclenchera un nouveau déploiement qui mettra à jour la couleur du site web.

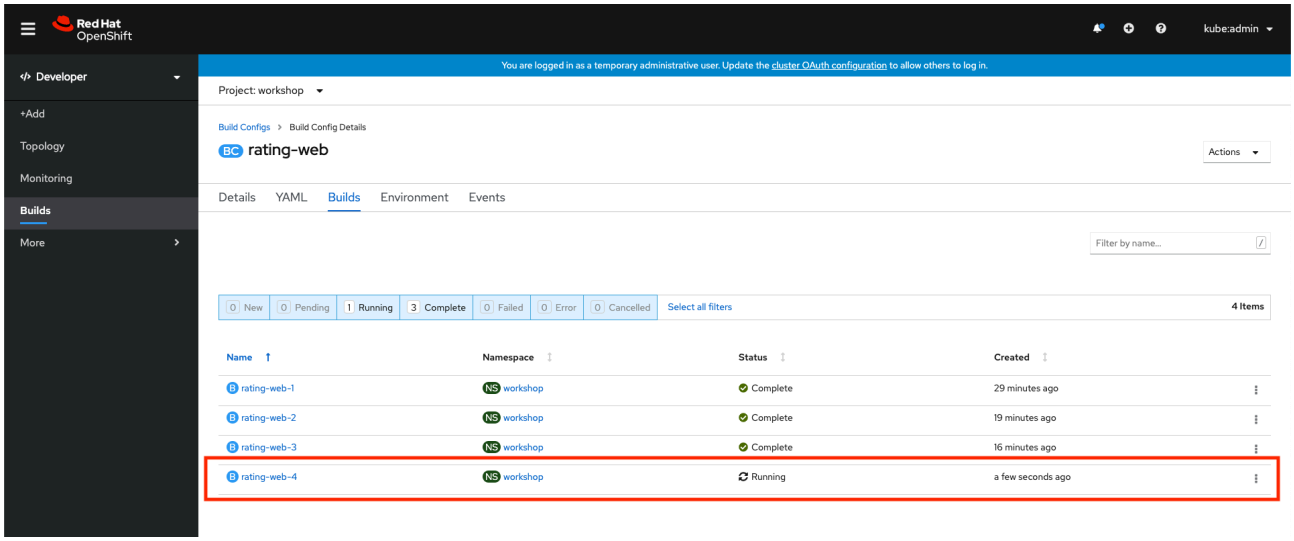


Figure 7.17 : Onglet Builds montrant une génération en cours

Retournez maintenant sur votre page `ratings-web`. Si tout s'est bien passé, vous verrez que la couleur de l'arrière-plan a changé !

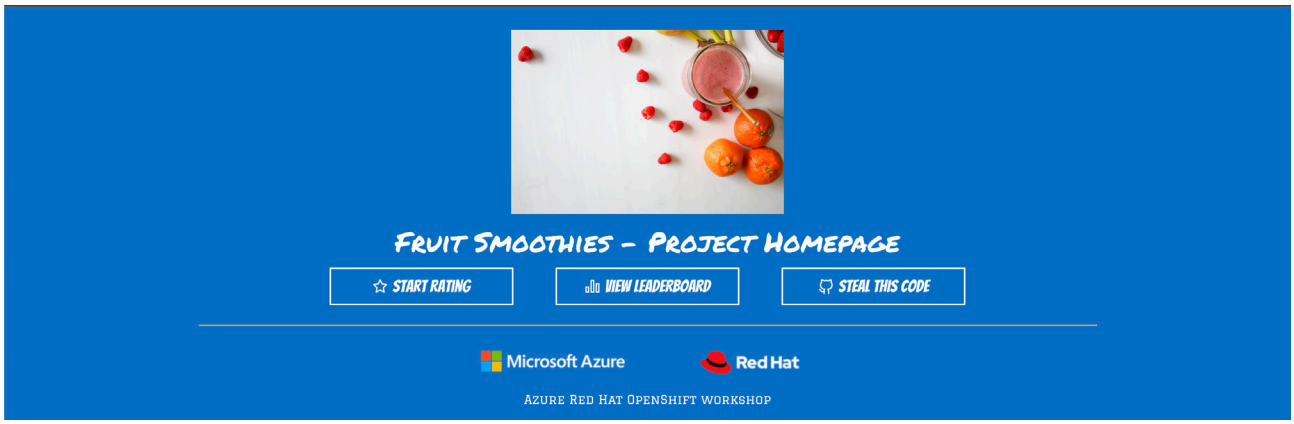


Figure 7.18 : Page d'accueil de Fruit Smoothies avec la nouvelle couleur

Synthèse

Dans ce chapitre, nous avons déployé une application de base comprenant trois applications de microservices plus petites : une base de données MongoDB, `ratings-api` et le code `ratings-web`. Bien que cette application diffère d'une application de production, elle permet de rappeler rapidement les concepts lors du déploiement d'applications sur Azure Red Hat OpenShift. Vous pouvez utiliser les instructions comme une base de référence pour déployer vos propres applications.

Red Hat OpenShift prend également en charge le déploiement d'applications à partir d'OperatorHub, ainsi que des charts Helm et des systèmes CI/CD externes tels qu'Azure DevOps. La stratégie de déploiement la plus adaptée à votre organisation dépend des outils et des technologies que vous utilisez en interne.

Dans le chapitre suivant, nous explorerons certaines des valeurs supplémentaires d'Azure Red Hat OpenShift : des valeurs qui distinguent OpenShift en tant que plateforme d'application, conçue sur Kubernetes. Nous examinerons les caractéristiques et les fonctions conçues pour répondre aux besoins complexes des applications d'entreprise.

Chapitre 8

Exploration de la plateforme d'applications

Dans les chapitres précédents, nous avons appris comment Red Hat OpenShift fournit une gamme de services conçus sur Kubernetes. Chacun de ces services se combine pour créer une véritable plateforme d'applications et appartient à l'une des cinq catégories suivantes : services de plateforme, services d'application, services de données, services de développement et services de cluster Kubernetes.

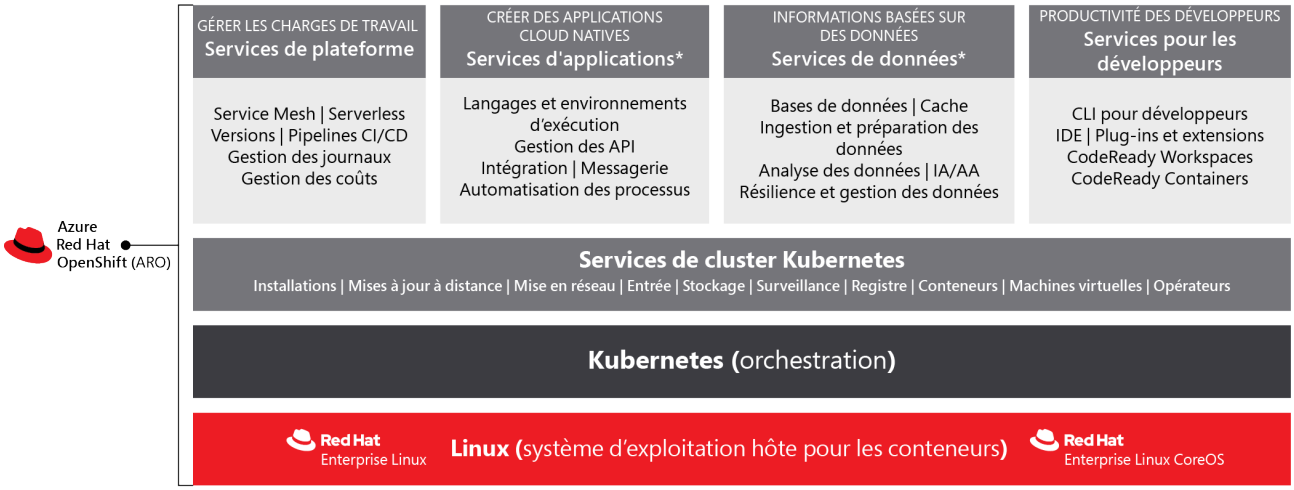


Figure 8.1 : Services inclus dans Azure Red Hat OpenShift

Les composants regroupés dans **OpenShift Platform Plus (Multicluster Management**, **Cluster Security** et **Global Registry**) sont des produits supplémentaires qui nécessitent une souscription. Ils sont compatibles avec Azure Red Hat OpenShift, mais ne sont pas inclus dans l'offre Azure Red Hat OpenShift.

Dans les sections suivantes, nous découvrirons certains des principaux avantages offerts par ces services. Vous trouverez également des liens vous permettant d'obtenir plus d'informations.

Services de clusters – registre intégré de conteneurs

Red Hat OpenShift inclut un registre de conteneurs interne et intégré dès le déploiement du cluster. Ce registre est utilisé à la fois pour les services internes au cluster, tels que les opérateurs de cluster, et par défaut pour les conteneurs d'applications des clients. Ce registre est standard et ne nécessite aucune configuration supplémentaire. Un opérateur d'infrastructure assure son bon fonctionnement.

Tous les clients qui déploient un service de conteneurs Kubernetes devront probablement déployer aussi leur propre registre de conteneurs pour préserver la confidentialité de leurs images de conteneurs. OpenShift supprime les processus d'installation et de configuration supplémentaires du jour 2, car le registre de conteneurs intégré est déjà disponible dans le cluster. Il s'agit d'un exemple de fonctionnalité simple qui permet de gagner du temps dans OpenShift.

Aperçu du registre intégré de la plateforme de conteneurs OpenShift

Un administrateur de cluster a souvent le choix d'exposer ce registre de conteneurs en dehors du cluster pour que les utilisateurs externes à OpenShift puissent pousser des images de conteneurs dans le registre. Azure Red Hat OpenShift prend cette fonctionnalité intégralement en charge. La procédure à appliquer figure dans la documentation standard d'OpenShift sur l'exposition du registre.

Services de plateforme – OpenShift Pipelines

Les clients de Red Hat OpenShift peuvent concevoir leurs applications de nombreuses façons. Ils disposent de nombreux outils CI/CD populaires, tels que Jenkins, CircleCI et GitHub Actions, ainsi que de plug-ins qui prennent en charge OpenShift. OpenShift fournit également des fonctionnalités sur la plateforme pour créer des applications à l'aide de pipelines de conteneurs cloud-natifs, via l'opérateur OpenShift Pipelines.

OpenShift Pipelines repose sur le projet communautaire Tekton. Chaque étape du pipeline, comme l'extraction du code d'un dépôt Git, l'exécution d'un compilateur Java ou l'assemblage d'un paquet RPM, est exécutée dans un conteneur. Cela signifie que les développeurs et les opérateurs peuvent former des pipelines complexes et sophistiqués en tirant profit de tous les avantages offerts par les conteneurs, pour concevoir des logiciels.

L'installation d'OpenShift Pipelines est possible via OperatorHub. Accédez à **OperatorHub** et sélectionnez l'opérateur pour commencer l'installation. Aucune configuration n'est nécessaire et l'installation de l'opérateur prend généralement moins d'une minute.

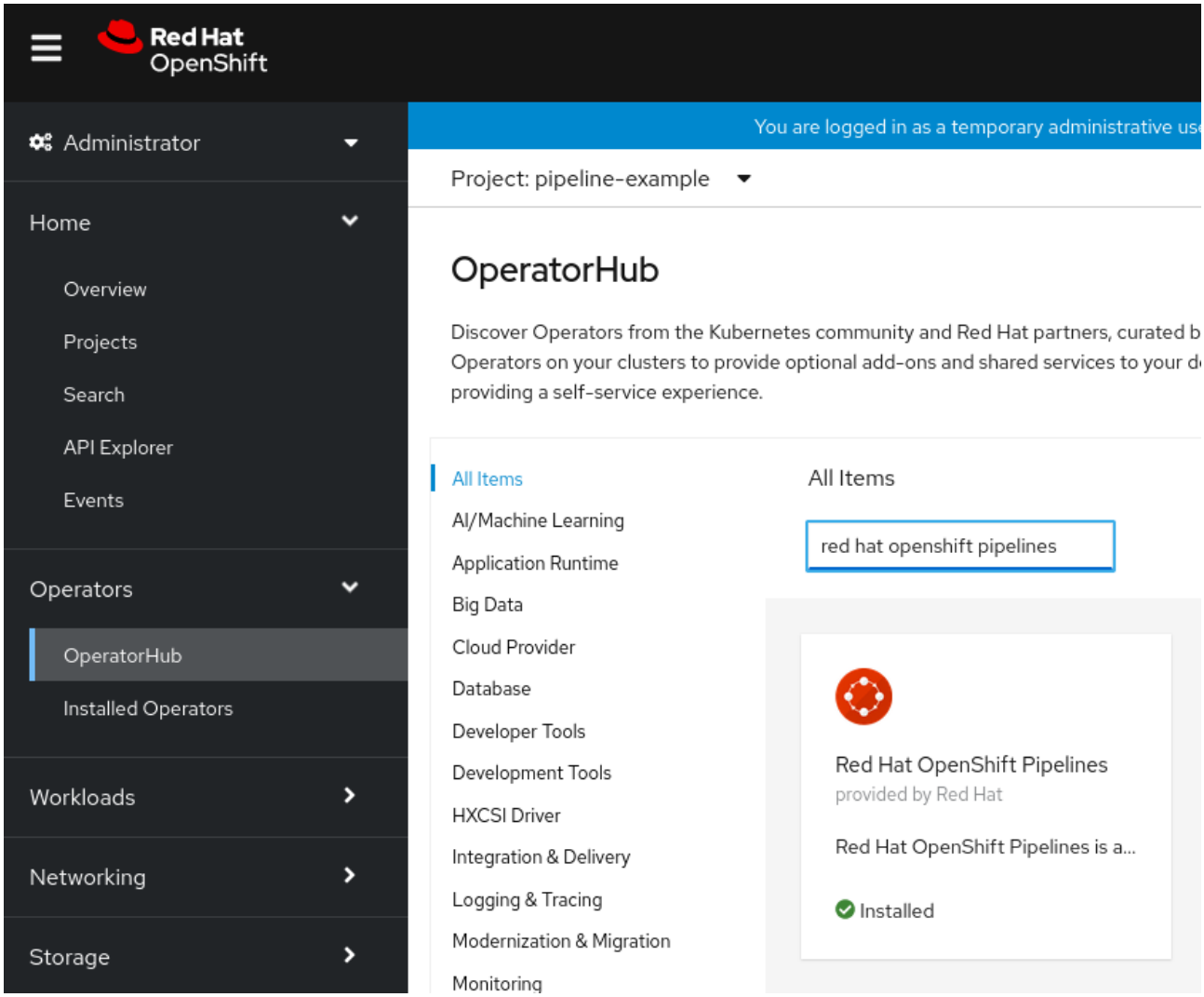


Figure 8.2 : Installation d'OpenShift Pipelines via OperatorHub

Une fois l'opérateur OpenShift Pipelines installé, une nouvelle section **Pipelines** s'affiche dans la barre latérale de même que l'option permettant d'ajouter des pipelines aux éléments du catalogue, comme lors de la création de l'exemple Node.js.

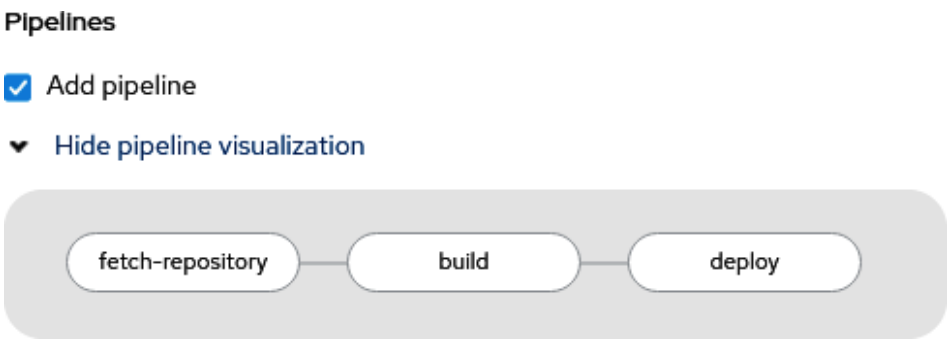


Figure 8.3 : Ajout de pipelines

OpenShift Pipelines permet l'utilisation de l'outil de création visuel pour configurer et créer des pipelines complexes et ramifiés. Voici un exemple de capture d'écran d'un pipeline plus complexe :

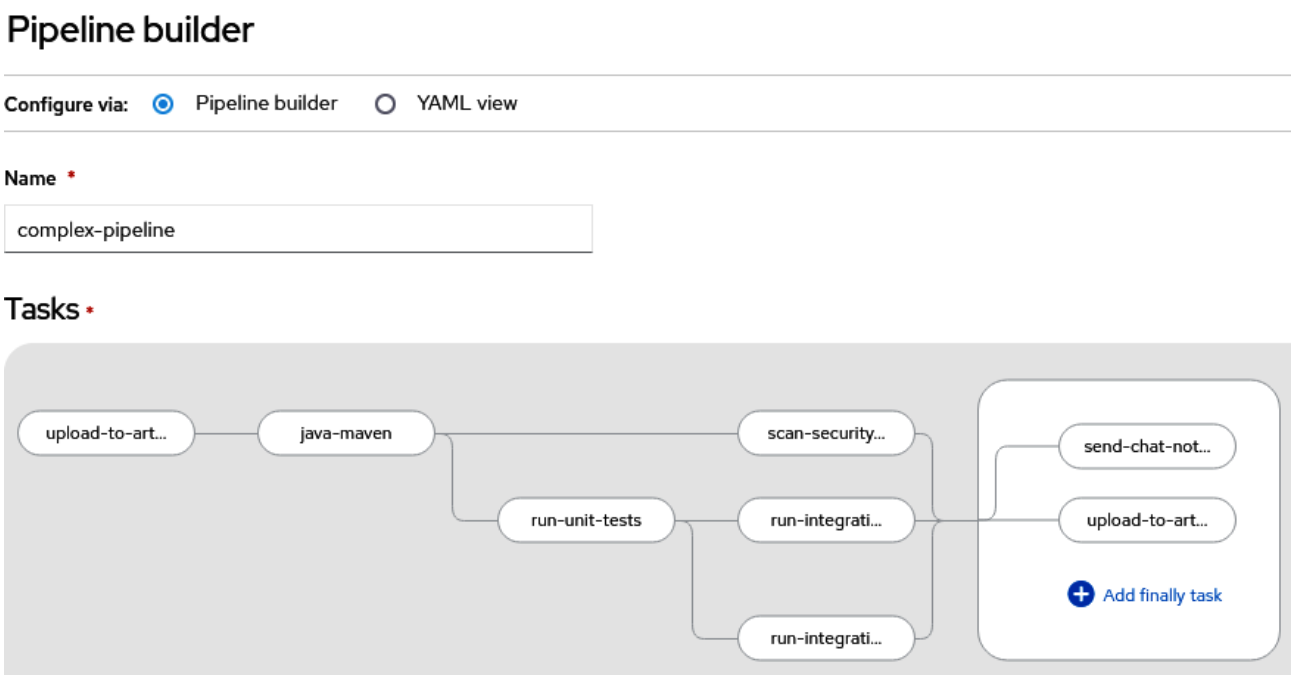


Figure 8.4 : Pipeline complexe

Alors que de nombreuses organisations utilisent toute une série d'outils et de technologies pour créer leurs logiciels, OpenShift Pipelines permet d'intégrer facilement la build directement dans OpenShift tout en tirant profit de tous les avantages offerts par les conteneurs. Vous disposez d'une solution CI/CD cohérente et facile à utiliser, qui ne nécessite qu'une configuration très minime, quelle que soit l'infrastructure sous-jacente utilisée.

Pour en savoir plus

- Comprendre OpenShift Pipelines dans OpenShift
- Site communautaire de Tekton

Services de plateforme – OpenShift Serverless

C'est une erreur de croire que les conteneurs ne sont qu'une technologie utile pour les services à long terme. En fait, de nombreuses tâches de courte durée et fonctions serverless s'exécutent sur la base de conteneurs de courte durée. En raison des avantages que les conteneurs offrent en termes de rapidité de démarrage, de cohérence et de facilité d'arrêt, ils conviennent également bien aux charges de travail serverless. Comme tous les services serverless requièrent des serveurs sous-jacents pour exécuter du code, ce modèle s'appelle aussi « fonction en tant que service ».

Red Hat OpenShift prend en charge des charges de travail serverless, ou fonction en tant que service, via l'opérateur OpenShift Serverless. Cet opérateur repose sur le projet open-source populaire Knative.

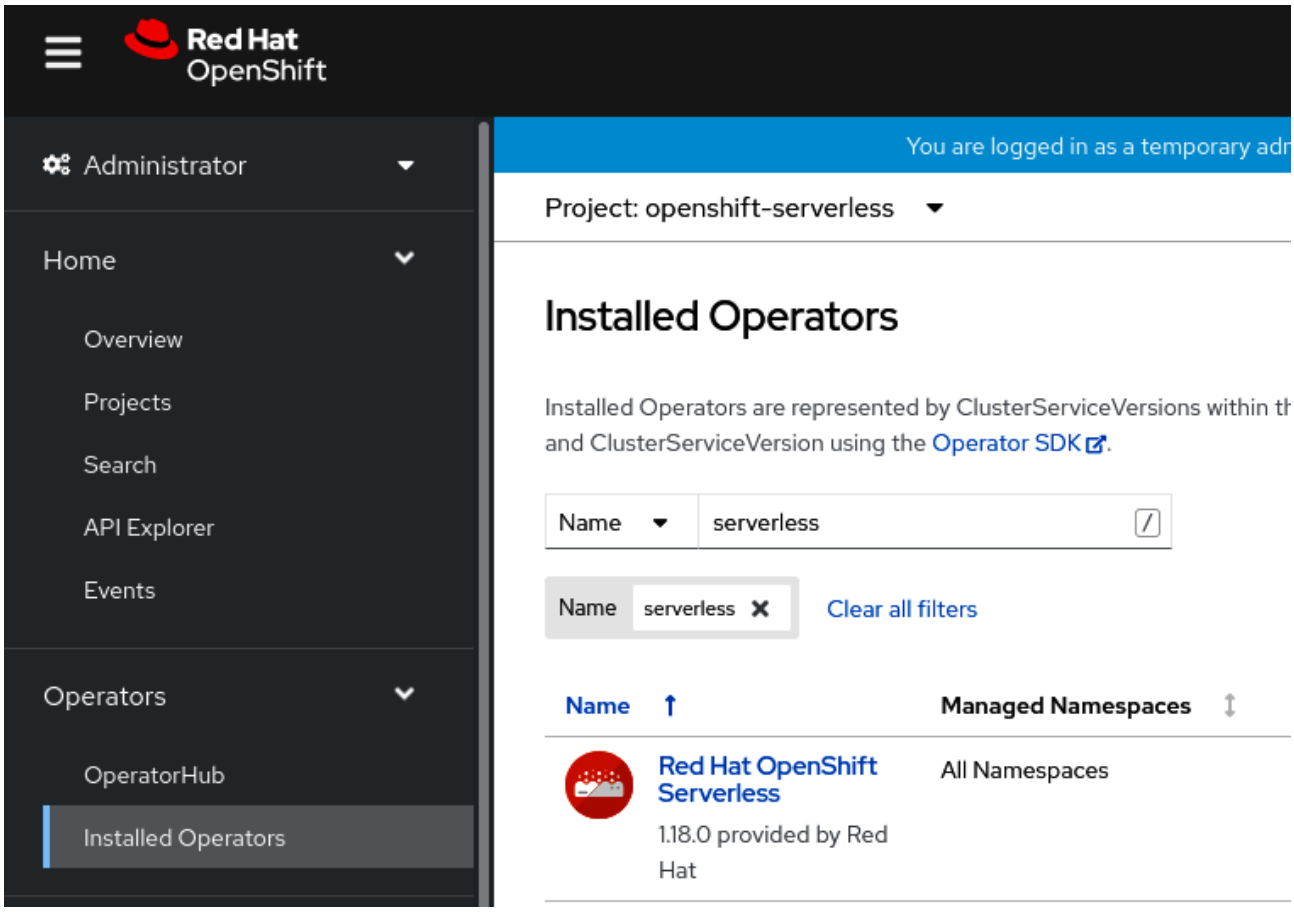


Figure 8.5 : Affichage des opérateurs installés

Une fois l'opérateur déployé sur le cluster (cette opération ne prend qu'une ou deux minutes), vous devez encore légèrement le configurer. Prêtez notamment attention à deux **définitions de ressources personnalisées (CRD)** : **Knative Serving** et **Knative Eventing**.

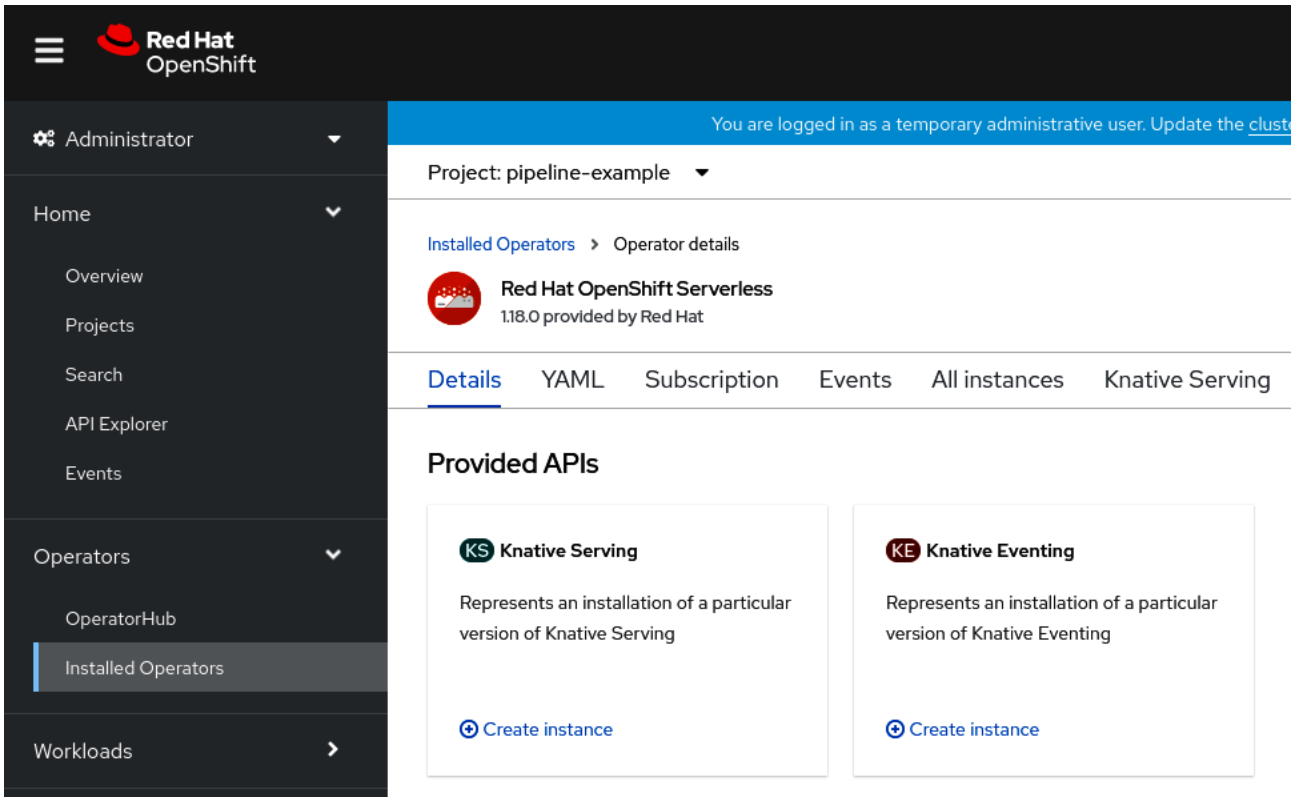


Figure 8.6 : Knative Serving et Knative Eventing dans le menu Operators (Opérateurs)

- **Knative Serving** simplifie le déploiement de l'application, évolue dynamiquement en fonction du trafic entrant et prend en charge des stratégies de déploiement personnalisées avec répartition du trafic ; par exemple, une fonction qui télécharge une image vers un panier de stockage et enregistre ce téléchargement dans une base de données NoSQL.
- **Knative Eventing** permet de « lier tardivement » les sources d'événements au moment de l'exécution de l'application (par opposition au moment de la création). Par exemple, une application qui répond aux nouveaux chargements d'images dans un panier de stockage. Il n'est pas nécessaire que cette application connaisse le panier de stockage au moment de la création ou du déploiement des événements. Knative Eventing permet de « lier » facilement cette source d'événements à l'application au moment de l'exécution.

Les deux sections suivantes proposent des exemples pour chacun de ces types d'applications serverless sur OpenShift.

Services de plateforme – OpenShift Serverless – Exemple : Knative Serving

Découvrons comment Knative Serving permet la mise à l'échelle automatique d'une application, notamment la mise à l'échelle à zéro en l'absence de demandes. Pour ce faire, nous pouvons utiliser une application très simple, une page web qui propose des images ASCII d'animaux domestiques :

- php-ascii-pets GitHub repository

L'ajout de ce dépôt à partir de Git est simple : Red Hat OpenShift détecte automatiquement une image de l'outil de création compatible de PHP.

OpenShift détecte automatiquement comment créer ce projet.

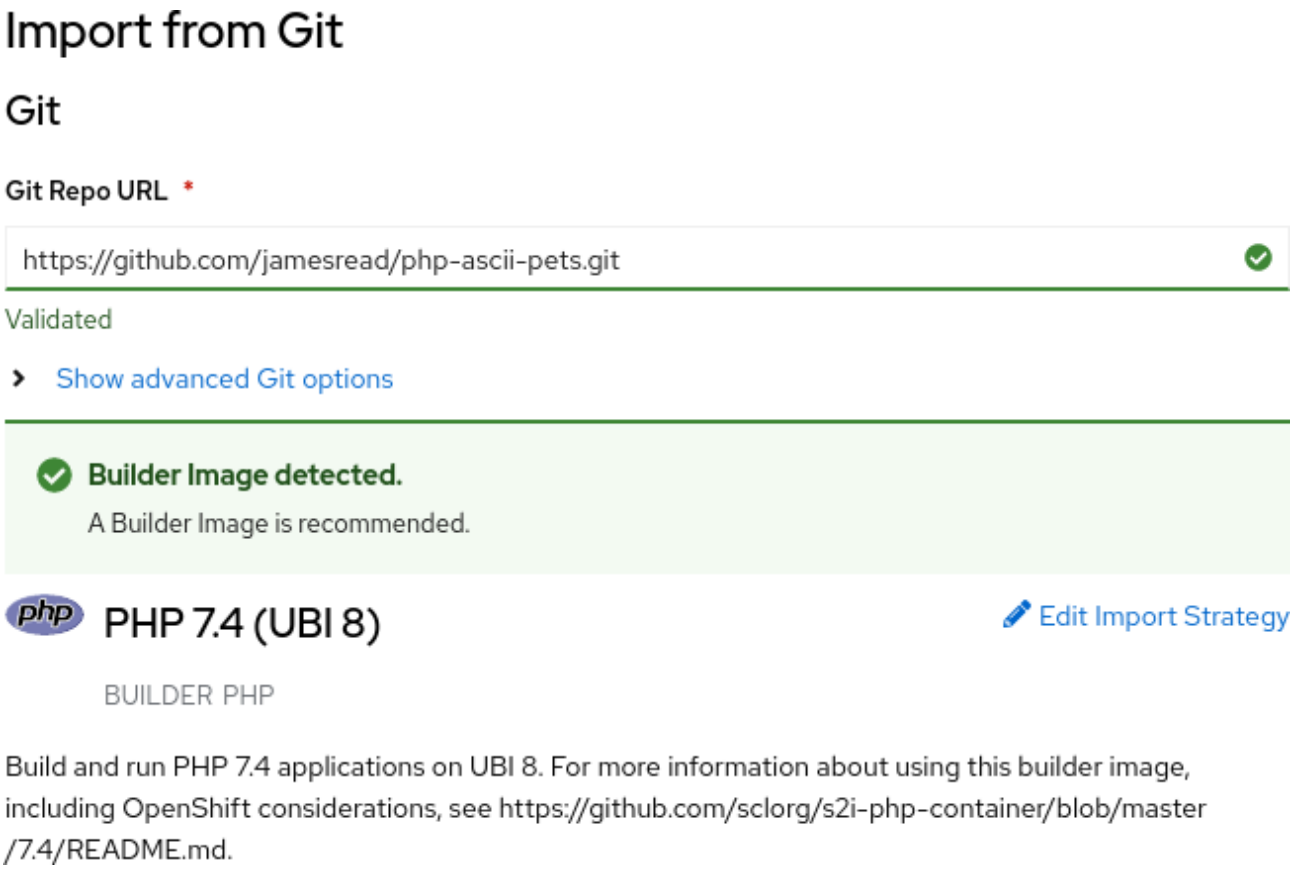


Figure 8.7 : Détection et recommandation automatiques d'une image de l'outil de création

C'est principalement le type de déploiement sélectionné qui fait de ce déploiement un déploiement « serverless ». Notez que lorsque OpenShift Serverless est installé, un déploiement serverless est disponible.

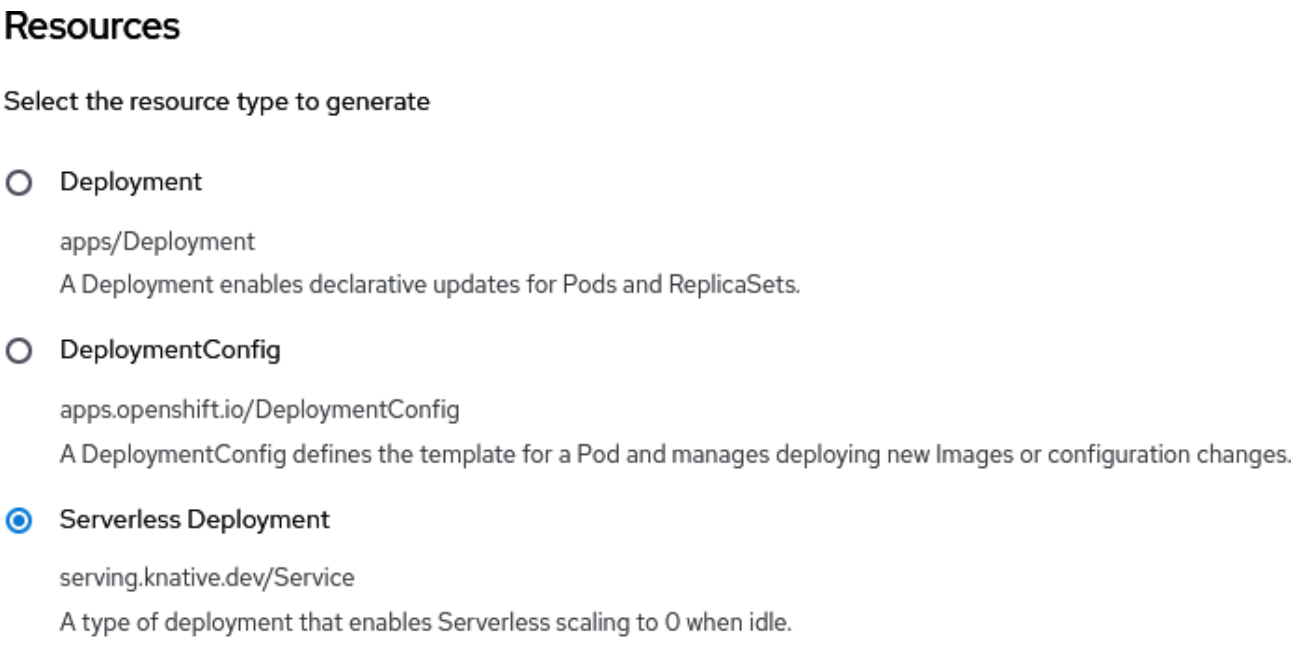


Figure 8.8 : Type de déploiement serverless

La première build s'achève après quelques instants. Une fois la build terminée, un pod devrait être déployé. La vue topologique devrait ressembler à ceci :

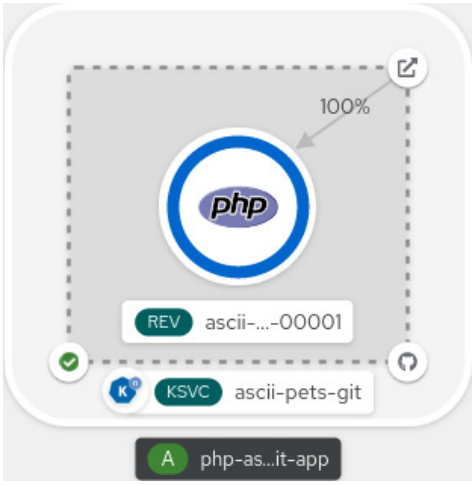


Figure 8.9 : Application Knative Serving

La page d'application actuelle ressemble à celle de la *figure 8.9*. Il s'agit d'une application très simple, mais l'animal de compagnie (dessin ASCII) est lié au nom d'hôte du pod. Si nous renforçons la charge supplémentaire dans notre application de démonstration simple, Knative Serving crée des pods supplémentaires. Dès lors vous pouvez voir visuellement que plusieurs animaux domestiques sont traités !

Toutefois, en l'absence de demande pendant une minute, Knative Serving remet l'application à zéro. La vue de la topologie montre que l'application n'est plus en cours d'exécution.

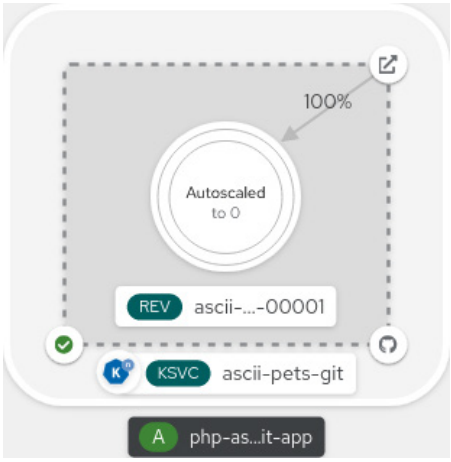


Figure 8.10 : Déploiement serverless avec Knative Serving et remise à zéro de l'application.

Enfin, si un tiers visite la page, l'application est automatiquement remise à l'échelle sur 1, 2, 3 répliques ou plus, selon le nombre d'instances qu'OpenShift estime nécessaire pour traiter la demande.

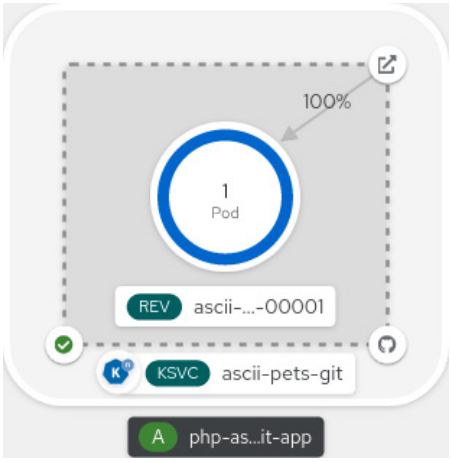


Figure 8.11 : Mise à l'échelle automatique en fonction du trafic

Red Hat OpenShift Serverless, avec Knative Serving, permet aux organisations d'adapter leur capacité de manière dynamique en minimisant la configuration supplémentaire et sans modifier les applications. Cela peut s'avérer particulièrement utile pour maintenir les déploiements à la taille appropriée tout en évitant d'utiliser et de payer des ressources inutilement.

Pour en savoir plus

- À propos d'OpenShift Serverless
- learn.openshift.com, qui comprend un cours sur OpenShift Serverless
- Site communautaire de Knative

Services de plateforme – OpenShift Serverless – Exemple : Knative Eventing

À partir de l'exemple d'application et des principes fondamentaux du chapitre précédent, OpenShift Serverless peut également faire évoluer une application en fonction d'autres paramètres que le trafic entrant. Dans des scénarios tels que l'architecture orientée événements notamment, il est souhaitable que vous puissiez faire évoluer les instances d'une application pour qu'elles traitent les événements provenant d'une file d'attente de messages ou se réveillent en fonction d'une minuterie.

La console OpenShift permet de configurer facilement différentes sources d'événements avec le même déploiement.

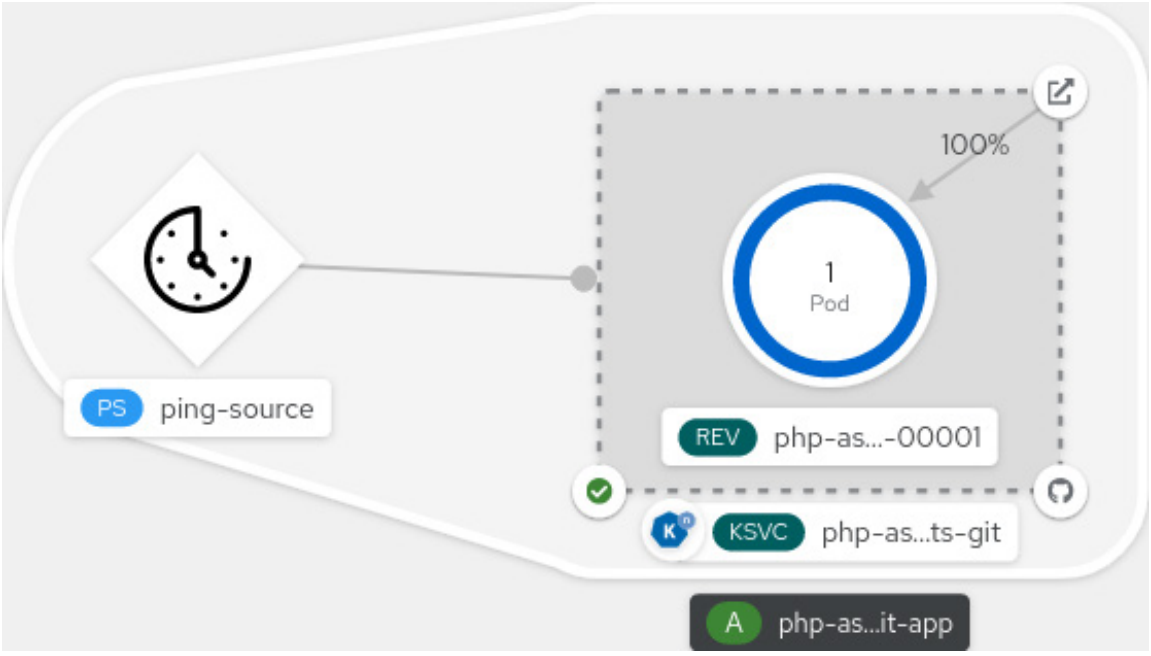


Figure 8.12 : Source d'événement « ping » qui envoie un ping à l'application en fonction d'une minuterie

Un cas d'utilisation simple pour une source d'événement ping consiste notamment à « réveiller » un conteneur de travail de sauvegarde chaque nuit à minuit. Vous pouvez également utiliser cette minuterie ping pour un conteneur de surveillance au fonctionnement périodique.

Pour en savoir plus

- OpenShift Serverless Eventing expliqué en 5 minutes
- À propos d'OpenShift Serverless

Services de plateforme – OpenShift Service Mesh

Red Hat OpenShift Service Mesh repose sur le projet open-source Istio. Ce service ajoute une couche transparente aux applications distribuées existantes sans demander de modification du code du service. Vous ajoutez la prise en charge de Red Hat OpenShift Service Mesh aux services en déployant un proxy sidecar spécial dans votre environnement. Ce dernier va intercepter toutes les communications réseau entre les microservices. Vous configurez et gérez le service mesh avec les fonctionnalités du plan de contrôle.

Cas d'utilisation que vous pouvez activer avec OpenShift Service Mesh :

- Chiffrement transparent de la communication entre les services, avec mTLS automatique
- Traitement de plusieurs versions d'un service et autorisation des tests A/B, par exemple
- Observation de la façon dont les microservices communiquent, avec Kiali
- Traçage des appels de service à service, avec Jaeger

Comme toutes les autres fonctionnalités de la plateforme OpenShift, Service Mesh est installé avec un opérateur Red Hat.

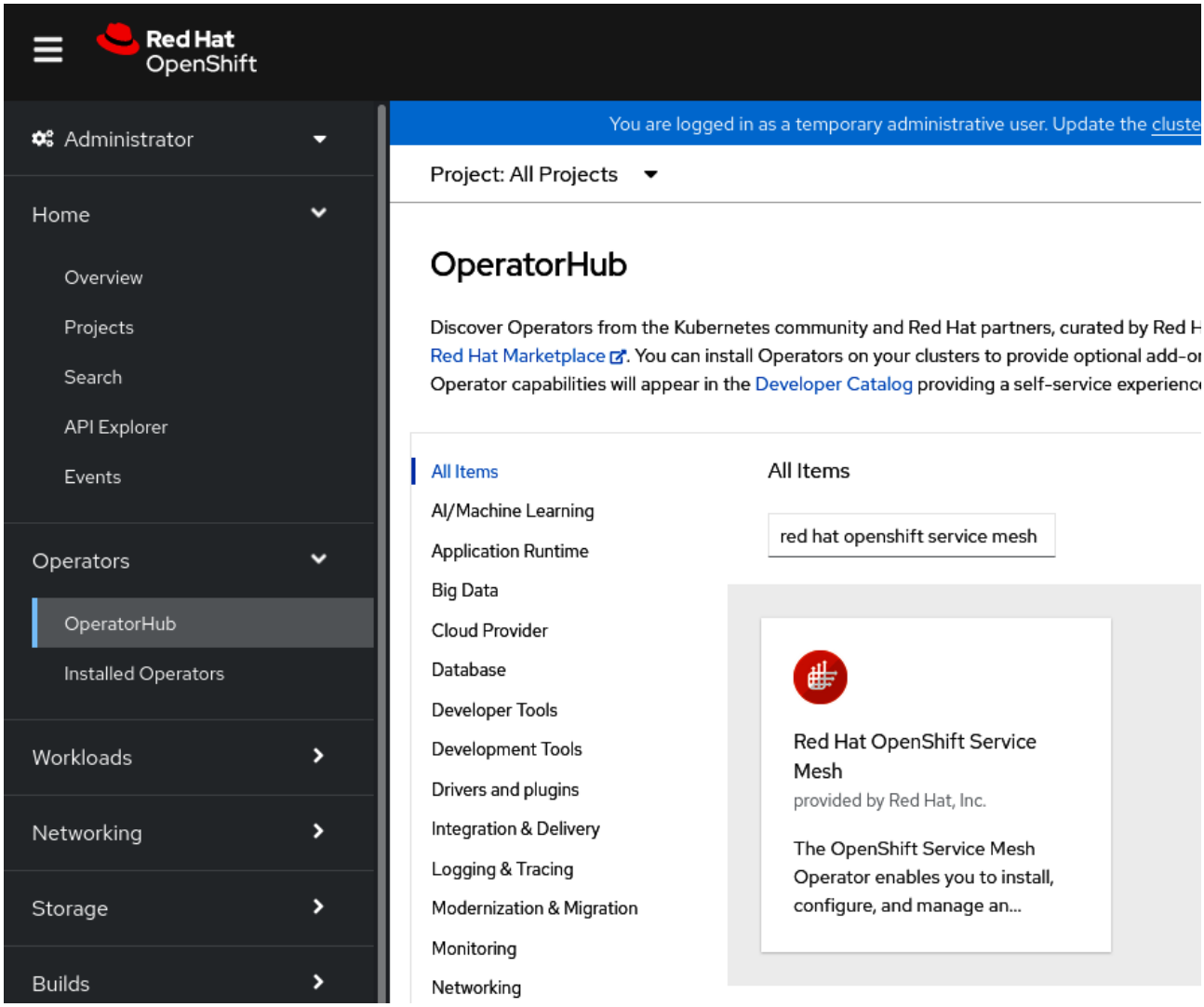


Figure 8.13 : Installation de Service Mesh via OperatorHub

La documentation d'OpenShift Service Mesh est accompagnée d'un excellent exemple d'application : BookInfo demo. Cette application simple émule une librairie, fonctionnant sur OpenShift.

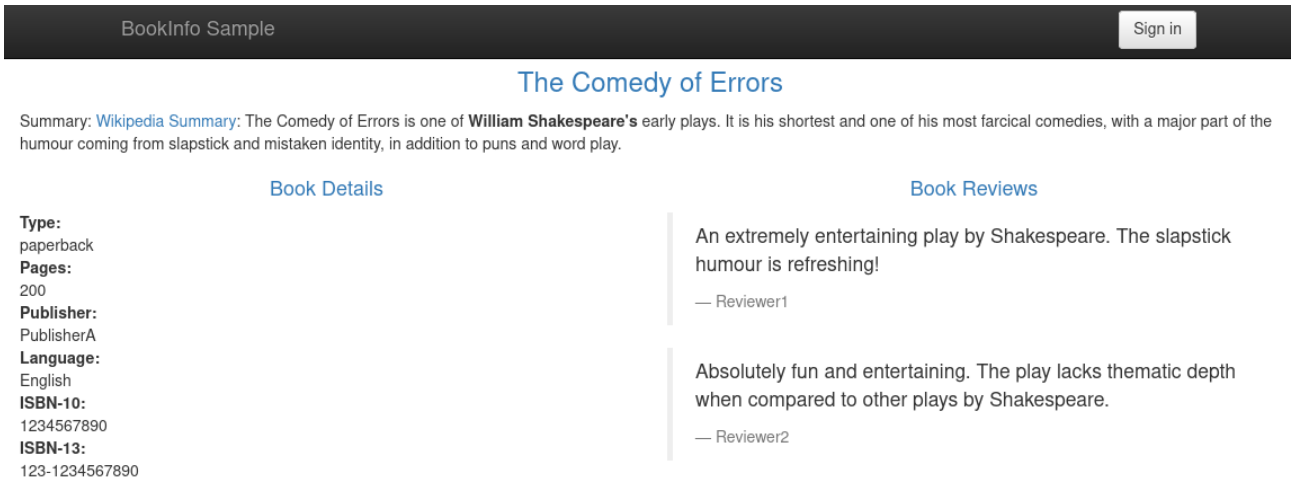


Figure 8.14 : Application BookInfo

Pour que l'application BookInfo fonctionne avec Service Mesh, vous devez créer un plan de contrôle Service Mesh. Vous pouvez le faire facilement via l'interface graphique d'OpenShift, comme illustré sur la *figure 8.15* :

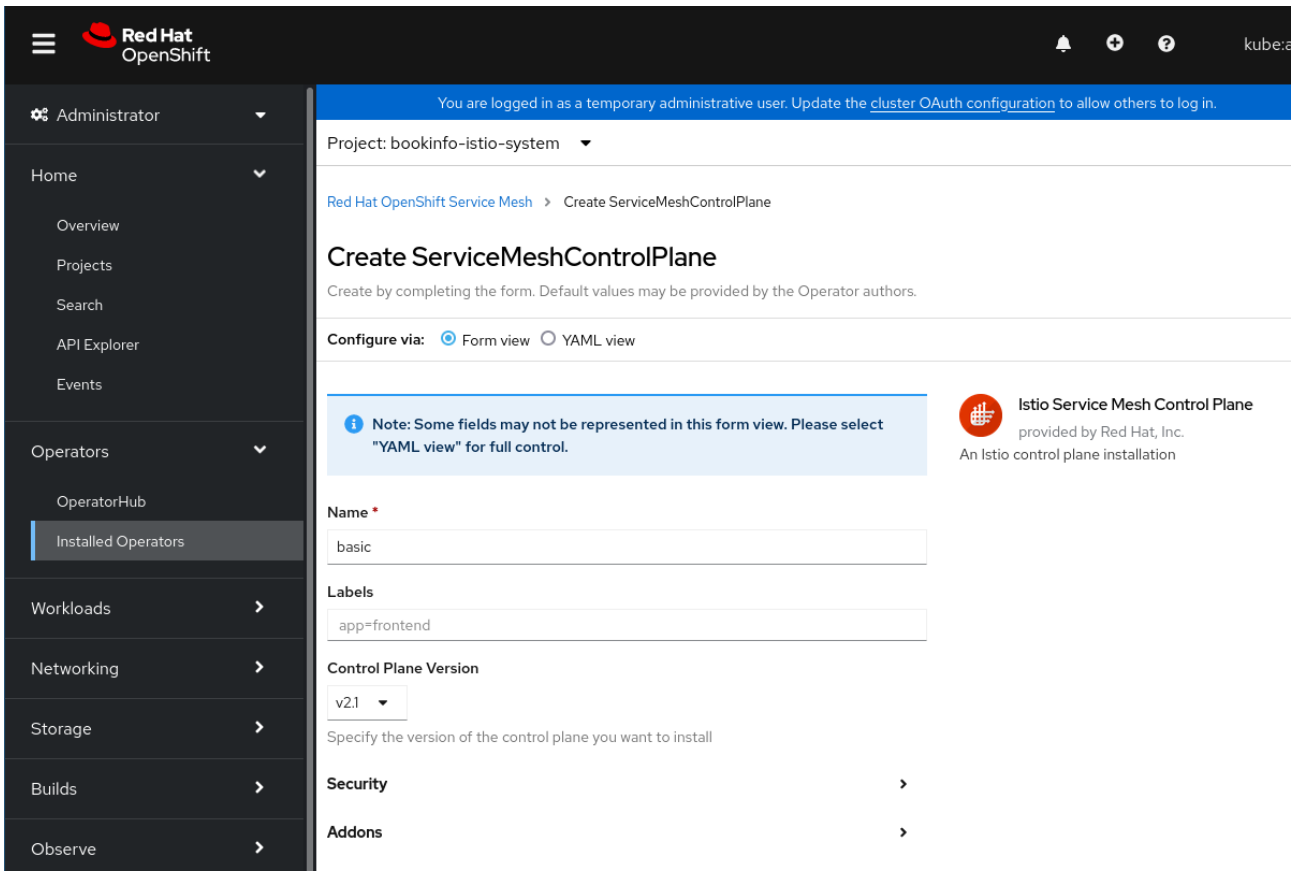


Figure 8.15 : Configuration du plan de contrôle dans le projet BookInfo bookinfo-istio-system

Le projet BookInfo acceptera le trafic entrant via le plan de contrôle. Ici, un projet distinct a été créé pour héberger le plan de contrôle : `bookinfo-istio-system`.

Vous n'avez pas besoin de séparer le plan de contrôle Service Mesh et votre projet. Vous pouvez même partager un plan de contrôle Service Mesh entre plusieurs projets.

Dans les deux prochaines sections, nous découvrirons deux cas d'utilisation de Service Mesh de manière beaucoup plus détaillée : l'observabilité et le traçage distribué.

Services de plateforme – OpenShift Service Mesh – Observabilité avec Kiali

En examinant les pods sous-jacents qui composent cette application, vous pouvez voir dans la vue de la topologie de Red Hat OpenShift qu'elle comprend six microservices.

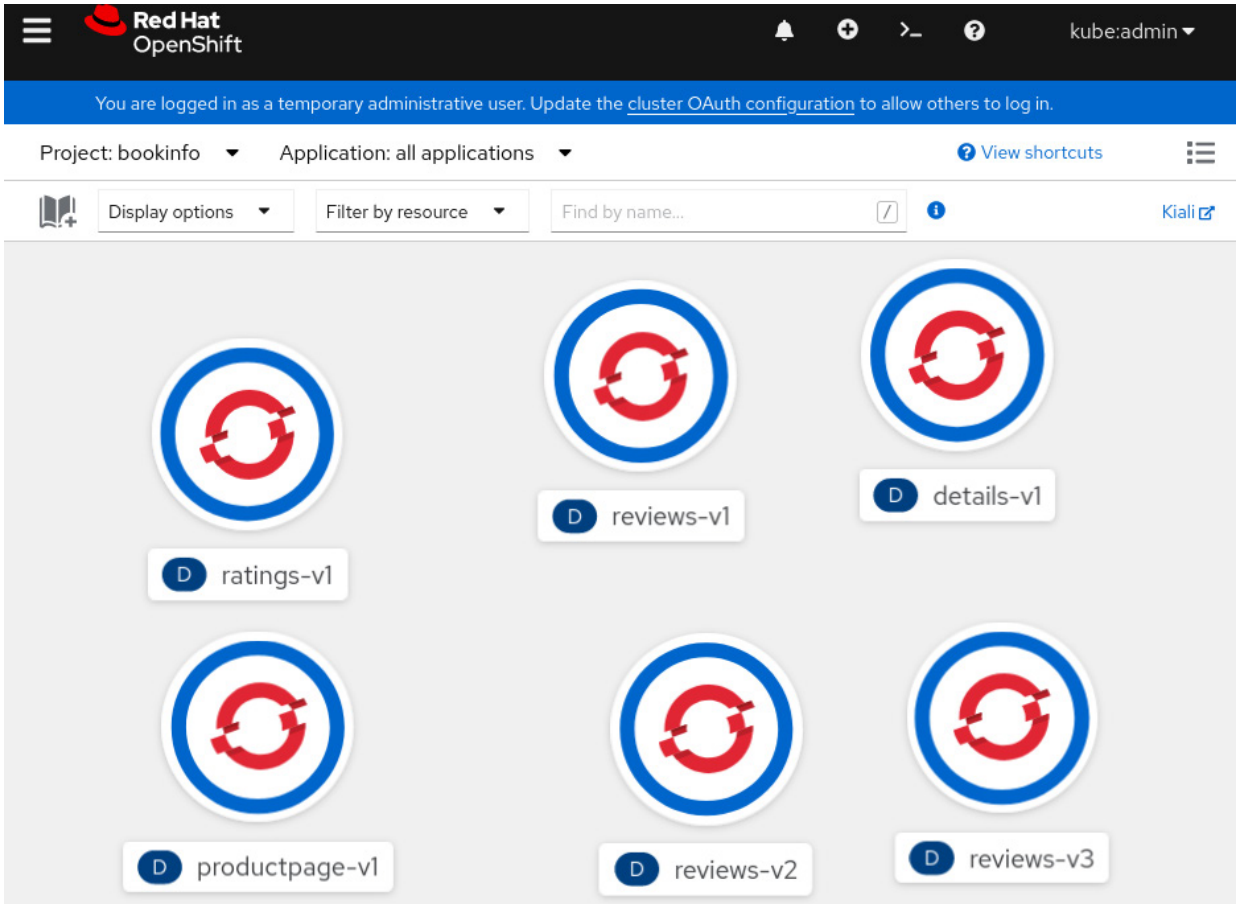


Figure 8.16 : Les six services qui constituent l'application BookInfo

Bien que cette vue soit utile, elle n'indique aucunement la manière dont ces services sont connectés. Découvrez le premier avantage de Service Mesh : l'observabilité. En ouvrant Kiali, une console légèrement différente fournie avec Service Mesh, nous constatons qu'il existe une représentation architecturale beaucoup plus détaillée de ces six mêmes services.

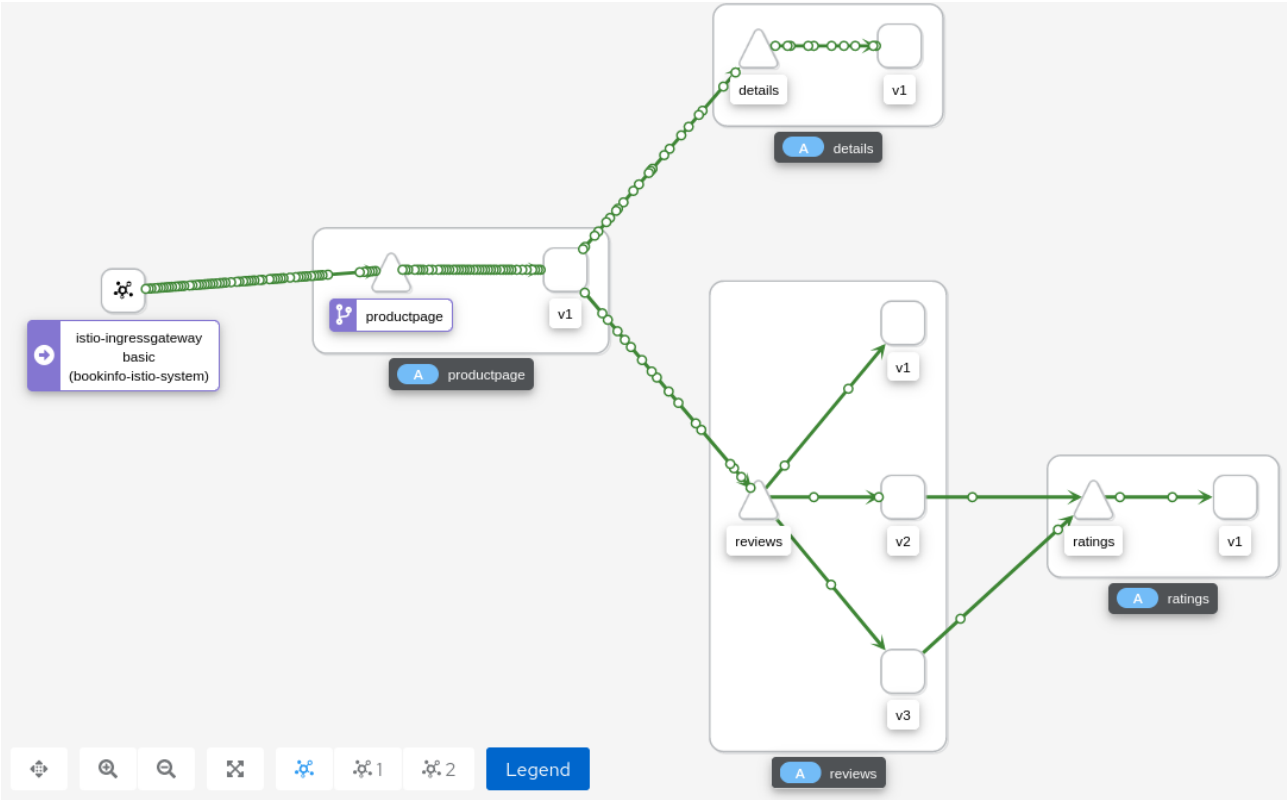


Figure 8.17 : Graphique d'architecture d'application généré automatiquement, activé par Service Mesh

Cette vue montre la connectivité réelle entre les services et un schéma de trafic en temps réel. Ici, l'application reçoit une quantité importante de trafic, chaque demande étant représentée par une animation en forme de cercle sur la connexion.

En poussant plus loin cette observabilité, un administrateur peut cliquer sur l'une des connexions de service et voir le profil du trafic. Ici, nous pouvons voir que l'état du trafic est principalement **HTTP OK**.

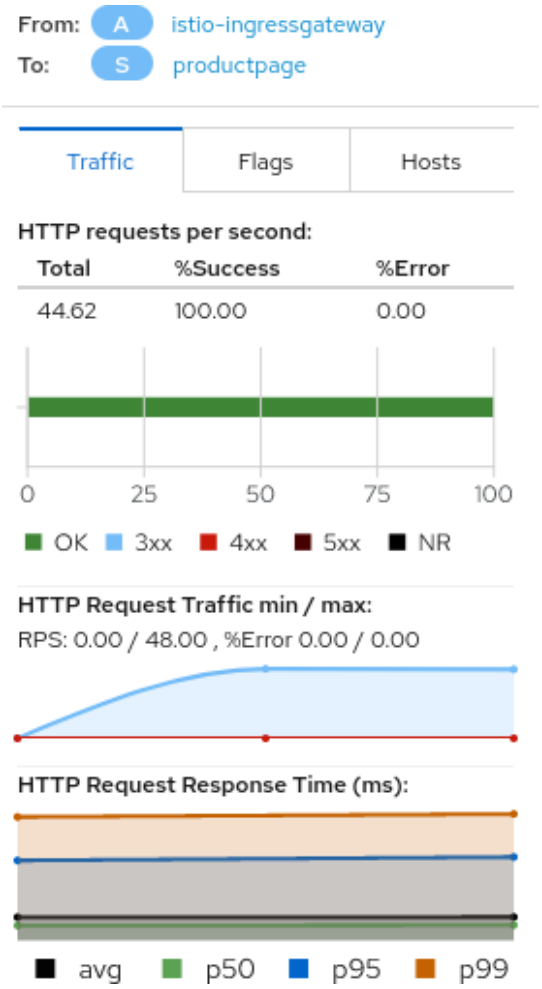


Figure 8.18 : Différents états HTTP

Nous pouvons introduire une erreur artificielle dans cette application en arrêtant le microservice `details`, en ajustant les réplicas sur zéro :

Figure 8.19 : Réplicas sur zéro dans le service « details » pour simuler une défaillance

En revenant maintenant à la vue Kiali, vous remarquerez l'ajout de quelques composants supplémentaires au schéma. Vous constaterez surtout que la connexion au service `details` est surlignée en rouge pour indiquer une erreur.

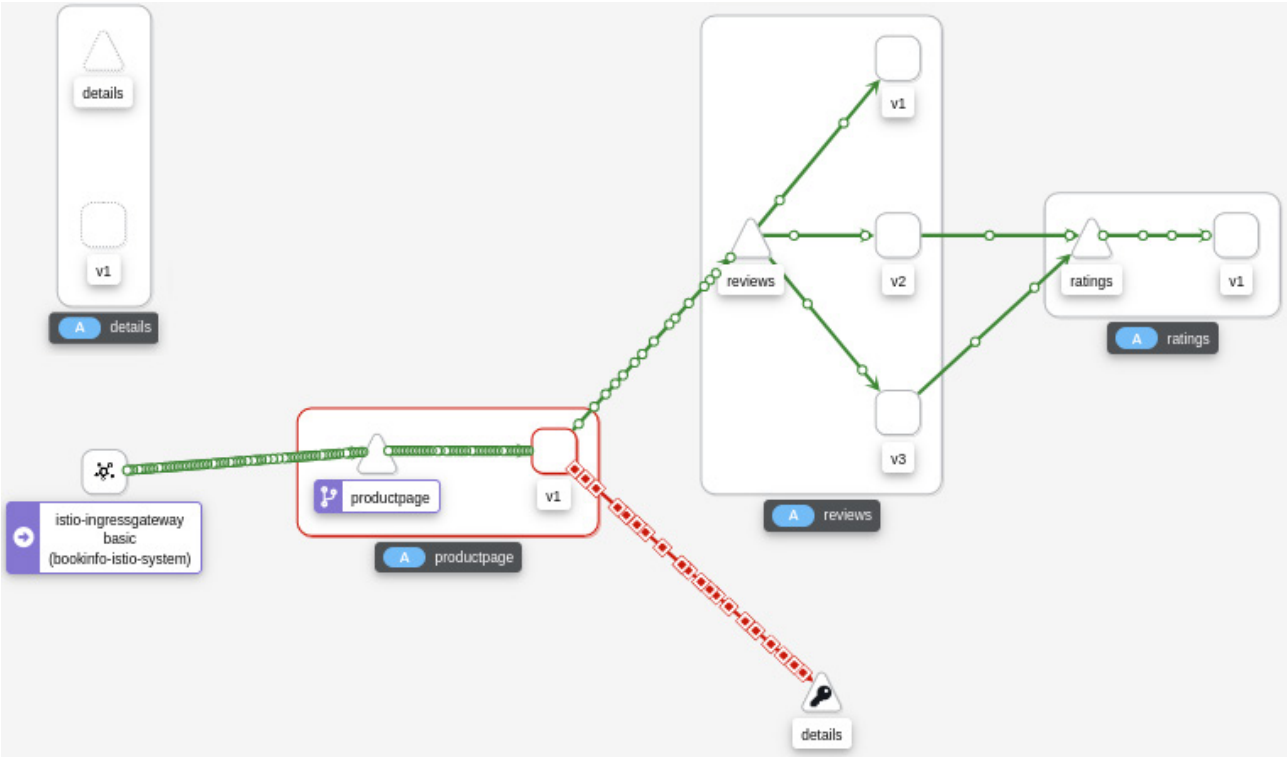


Figure 8.20 : Connexion rouge indiquant un problème avec ce service

Nous avons appris que Kiali, dans Service Mesh, facilite vraiment l'observabilité. Service Mesh est utile dans les petites applications comme BookInfo, mais à mesure que les applications deviennent plus imposantes et plus complexes, impliquant parfois 20 microservices ou plus et de nombreux types d'interactions différents, disposer d'outils tels que Service Mesh et Kiali devient incroyablement précieux, voire essentiel.

Services de plateforme – Red Hat OpenShift Service Mesh – Traçage distribué avec Jaeger

Jaeger est un autre service extrêmement précieux fourni par Service Mesh. Il permet un traçage distribué des applications de microservices complexes. Dans la section précédente, nous avons examiné l'application BookInfo et constaté que la mise hors ligne du service `details` entraînait l'échec des demandes.

Ici, la cause de l'échec était simple à identifier : le service `details` était indisponible par notre faute ! Face à une cause plus mystérieuse nécessitant une enquête plus approfondie, le traçage distribué de Jaeger s'avère néanmoins utile.

Jaeger suit les demandes du premier service jusqu'aux connexions ultérieures dans le système. Bien qu'un code d'application supplémentaire soit nécessaire pour activer Jaeger, les bibliothèques clientes et les modifications minimales du code rendent cette opération relativement facile pour les développeurs.

En examinant le service `productpage` (écrit en Python), nous pouvons repérer le code pertinent qui permet le traçage distribué par Jaeger dans ce service. Ce code importe la bibliothèque client Jaeger dans le service `productpage` :

```
from jaeger_client import Tracer, ConstSampler
from jaeger_client.reporter import NullReporter
from jaeger_client.codecs import B3Codec
```

Une fois la bibliothèque cliente importée, la page du produit doit configurer un nouveau traceur. Ce code initialise un traceur Jaeger `Tracer` dans le service `productpage` :

```
tracer = Tracer(
    one_span_per_rpc=True,
    service_name='productpage',
    reporter=NullReporter(),
    sampler=ConstSampler(decision=True),
    extra_codecs={Format.HTTP_HEADERS: B3Codec()}
)
```

Enfin, toujours dans le service `productpage`, nous indiquons à Jaeger que nous voulons créer un nouvel intervalle via une nouvelle requête vers plusieurs services :

```
span = tracer.start_span(
    operation_name='op', child_of=span_ctx, tags=rpc_tag
)
```

Jaeger va suivre ce traceur à travers plusieurs services. Ces derniers requièrent également des adaptations pour fonctionner avec Jaeger, mais des bibliothèques clientes sont disponibles pour la plupart des langages de programmation courants.

Le code source complet du service `productpage` figure sur GitHub.

Deux options permettent d'instrumenter le code : utiliser les bibliothèques clientes de Jaeger, désormais considérées comme obsolètes, ou utiliser de préférence les options standard ouvertes du projet OpenTelemetry :

- Bibliothèques OpenTelemetry

Une fois cet effort d'instrumentation d'une application terminé, des traces ressemblant à celles-ci sont visibles :

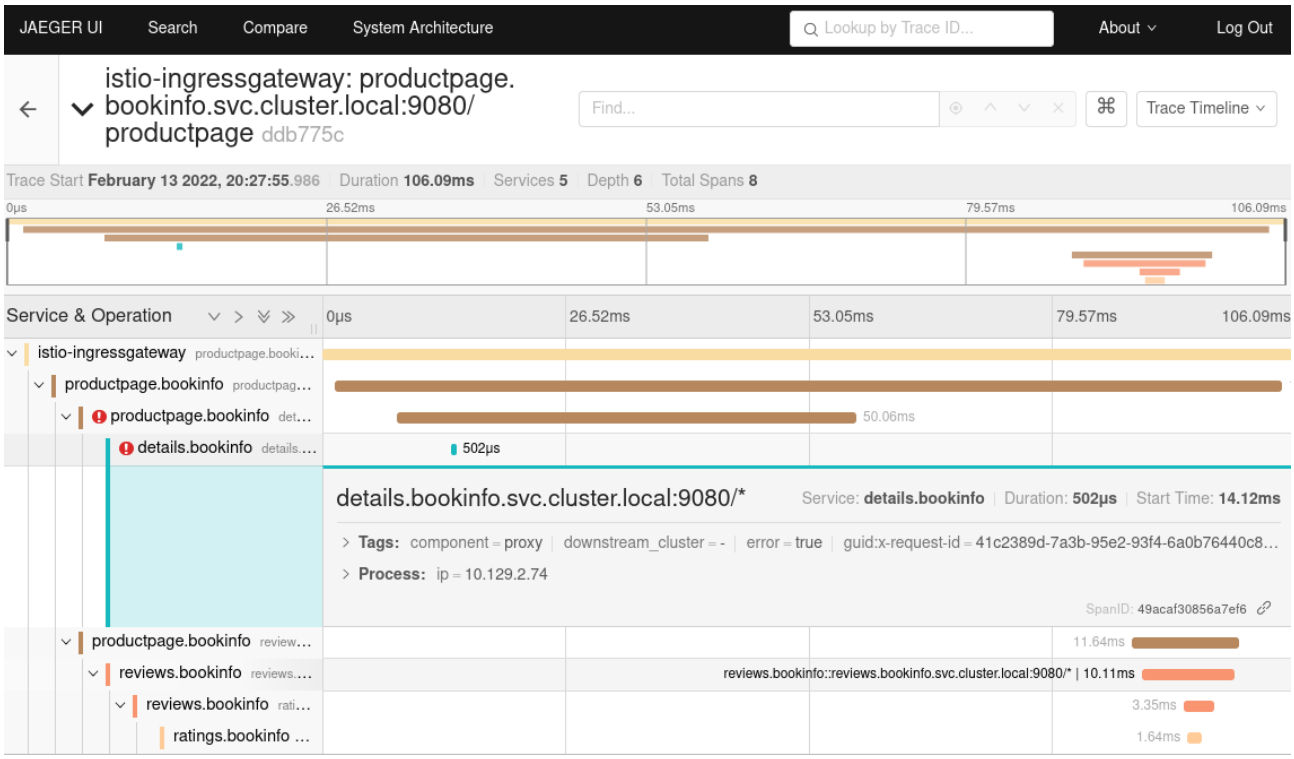


Figure 8.21 : Trace d'un appel

Cette vue montre la trace d'un appel, un peu comme Kiali, mais ici dans une vue hiérarchique. Vous pouvez développer chacune des lignes affichées pour obtenir les détails relatifs à la durée de la demande, l'heure de début, l'adresse IP source, etc. Là encore, ce niveau d'informations devient presque essentiel lorsqu'il s'agit d'applications de microservices complexes.

Concernant l'erreur que nous essayions de diagnostiquer, la vue de la trace montre clairement que le microservice `details` rencontre des problèmes. Ici, l'erreur est simple, mais l'affichage de la vue détaillée montre que le service émet des codes d'erreur `HTTP 503`.

Figure 8.22 : Détails de l'erreur

HTTP 503 est le code d'erreur standard pour désigner un « service indisponible ». Ici, le problème est simple : le service est défini sur zéro réplica. La remise à l'échelle du service restaure ce dernier.

Jaeger et Kiali combinés sont des outils puissants à mesure que les applications de microservices deviennent plus complexes. Ils sont fournis dans le cadre du service Azure Red Hat OpenShift et ne nécessitent ni dépenses ni souscriptions supplémentaires.

Synthèse

Ce chapitre a abordé quelques-uns des principaux services à valeur ajoutée que la plateforme d'applications Red Hat OpenShift propose par opposition à un simple environnement Kubernetes « vide ». Même si vous pouvez reproduire un niveau de fonctionnalité similaire à celui d'OpenShift en installant tous les projets open-source en amont respectifs sur OpenShift, Azure Red Hat OpenShift est complexe du fait de l'intégration, du cycle de vie et de l'assistance qui l'accompagnent.

Ces différents services de plateforme, d'applications, de données, de développement et de cluster accroissent la productivité de vos développeurs et opérateurs lorsqu'ils utilisent Kubernetes et lorsqu'ils déploient plusieurs applications d'entreprise complexes.

Chapitre 9

Intégration à d'autres services

Il est tout à fait normal qu'une application ne puisse pas exister entièrement dans Red Hat OpenShift. La plupart des applications reposent sur des bases de données externes ou des services sur Azure sous une forme ou une autre.

Azure Red Hat OpenShift n'« interrompt » aucun type de connectivité ici. Si une application repose sur Azure CosmosDB, par exemple, elle devrait toujours pouvoir s'y connecter à partir d'Azure Red Hat OpenShift sans aucune modification de l'application. En fonction de votre application et de votre organisation, vous pouvez déployer certaines de ces dépendances externes séparément à partir d'Azure Red Hat OpenShift, ou simultanément.

Si vous pensez qu'il serait bénéfique de déployer ces dépendances externes avec votre application, Azure Service Operator peut grandement simplifier ce processus. Au lieu de devoir utiliser l'interface en ligne de commande Azure, Azure Cloud Shell ou les modèles ARM, vous pouvez déployer ces ressources comme si elles étaient natives dans Kubernetes avec Azure Service Operator.

Azure Service Operator

Azure Service Operator est un autre opérateur qui facilite la vie des développeurs et des administrateurs qui déploient des applications sur Azure Red Hat OpenShift. Entre autres avantages, il permet de provisionner facilement les services Azure sans devoir quitter la console OpenShift. Vous pouvez ainsi déployer plus rapidement et plus facilement les applications qui peuvent dépendre des services Azure, comme Azure CosmosDB.

Autre avantage, peut-être plus important, de l'utilisation d'Azure Service Operator : il permet de stocker ces dépendances aux services Azure parallèlement à la définition de l'application OpenShift, dans une approche de type infrastructure en tant que code, au moyen de fichiers YAML Kubernetes standard.

Le fonctionnement est simple : Azure Service Operator recherche les nouveaux **CRD**, définis en YAML, qui correspondent à la définition d'une ressource sur Azure. Azure Service Operator traduit ensuite ce YAML en appels d'API Azure nécessaires pour satisfaire la demande sur Azure. Une fois l'opérateur installé, les utilisateurs peuvent parcourir le catalogue OpenShift Developer et consulter l'écran de création de nombreuses ressources Azure courantes, comme les adresses IP publiques Azure, les bases de données Azure SQL ou le pare-feu Azure.

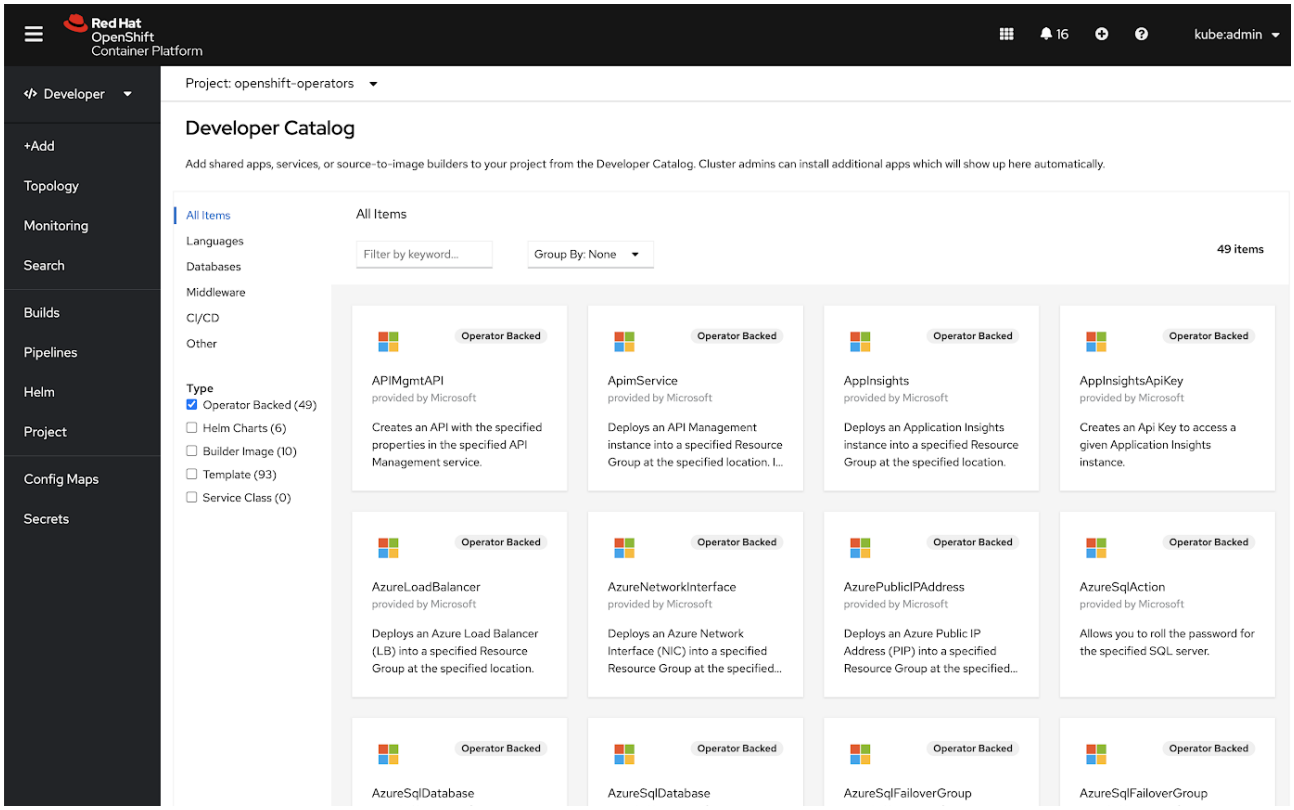


Figure 9.1 : Quelques-uns des services Azure exposés par Azure Service Operator

Vous pouvez installer Azure Service Operator via OperatorHub et le mettre à disposition de tous les utilisateurs d'OpenShift.

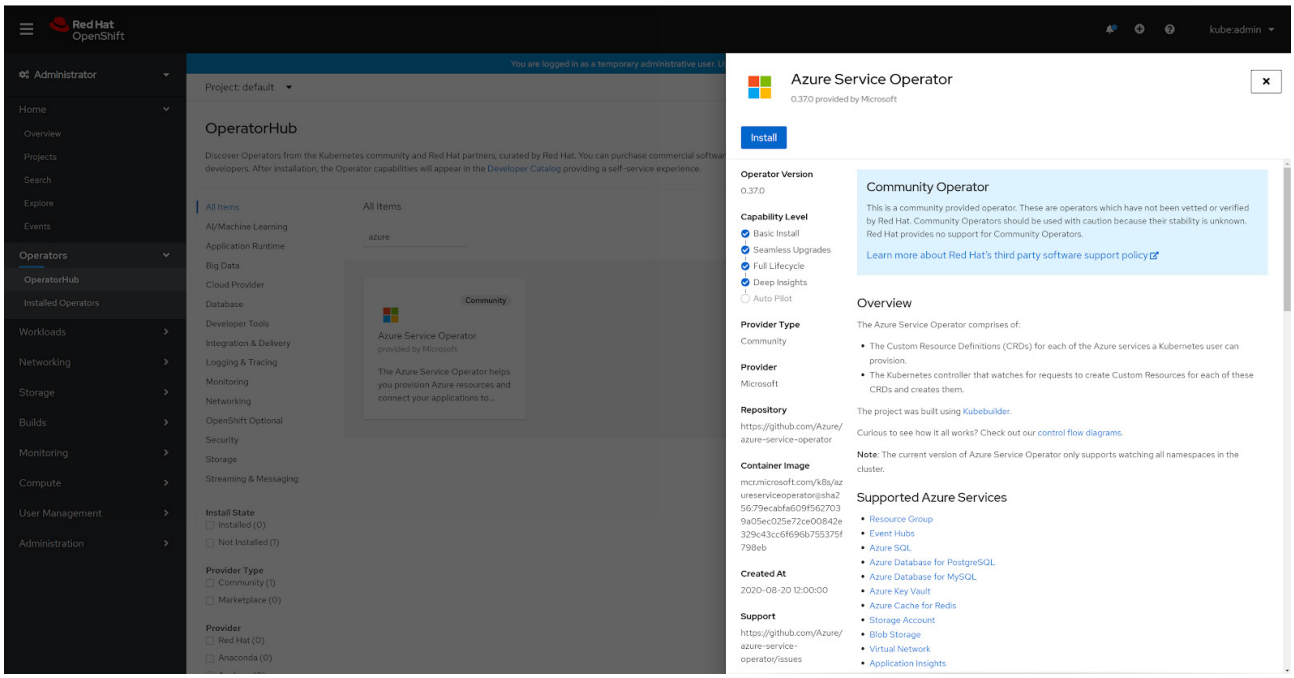


Figure 9.2 : Écran d'installation d'Azure Service Operator présentant la galerie OperatorHub en arrière-plan

L'utilisation d'Azure Service Operator n'est pas obligatoire pour les services fonctionnant sur Azure. Il est important de comprendre que les services Azure sous-jacents sont déployés nativement sur Azure et non sur OpenShift. Toutefois, grâce à Azure Service Operator, cette opération est plus rapide et plus facile pour les applications qui dépendent des services Azure.

Un cas d'utilisation populaire d'Azure Service Operator est le déploiement de bases de données dépendantes, en même temps que l'application. Par exemple, pour une application web qui utilise une instance d'Azure CosmosDB, déployez Azure CosmosDB avec Azure Service Operator dans le cadre du déploiement de l'application sur OpenShift. Azure Service Operator prend en charge Azure SQL Database, Azure Database for MySQL, Azure PostgreSQL, etc.

En partant du principe qu'Azure Service Operator est installé, vous pouvez stocker le manifeste Kubernetes suivant avec l'application OpenShift et l'utiliser pour provisionner une base de données MySQL :

Source : Extrait du dépôt d'échantillons d'Azure Service Operator.

```
apiVersion: azure.microsoft.com/v1alpha1
kind: MySQLServer
metadata:
  name: aso-wpdemo-mysqlserver
spec:
  location: eastus2
  resourceGroup: aso-wpdemo-rg
  serverVersion: "8.0"
  sslEnforcement: Disabled
  minimalTLSVersion: TLS10 # Possible values include: 'TLS10', 'TLS11', 'TLS12', 'Disabled'
  infrastructureEncryption: Enabled # Possible values include: Enabled, Disabled
  createMode: Default # Possible values include: Default, Replica, PointInTimeRestore (not
implemented), GeoRestore (not implemented)
  sku:
    name: GP_Gen5_4 # tier + family + cores eg. - B_Gen4_1, GP_Gen5_4
    tier: GeneralPurpose # possible values - 'Basic', 'GeneralPurpose', 'MemoryOptimized'
    family: Gen5
    size: "51200"
    capacity: 4
```

L'utilisation d'Azure Service Operator pour des services impliquant de nombreux services Azure avec des dépendances complexes est rare. Des outils tels que les modèles Azure ARM ou Bicep peuvent s'avérer plus adaptés le cas échéant.

Pour une introduction approfondie à Azure Service Operator, consultez les articles de blog suivants :

- Article de blog – Septembre 2020
- Azure Service Operator sur OperatorHub.io
- Azure Service Operator sur GitHub

Intégration à Azure DevOps

De nombreux utilisateurs souhaiteront intégrer Azure Red Hat OpenShift à Azure DevOps avec OpenShift Pipelines. Ces solutions peuvent coexister : certains projets utilisent une solution, d'autres une autre ou parfois, les projets utilisent les deux ! La décision d'utiliser l'une ou l'autre solution dépend de quelques facteurs.

De manière générale, Azure DevOps offre un haut niveau d'intégration aux autres outils Azure. Vous pouvez facilement déployer cette solution sur OpenShift et d'autres ressources de calcul.

D'autre part, OpenShift Pipelines est une offre bien intégrée qui accompagne OpenShift et assure une expérience multicloud cohérente.

Azure DevOps traite simplement OpenShift comme n'importe quel autre cluster Kubernetes. Par conséquent, toutes les interfaces et API standard de Kubernetes fonctionneront comme prévu.

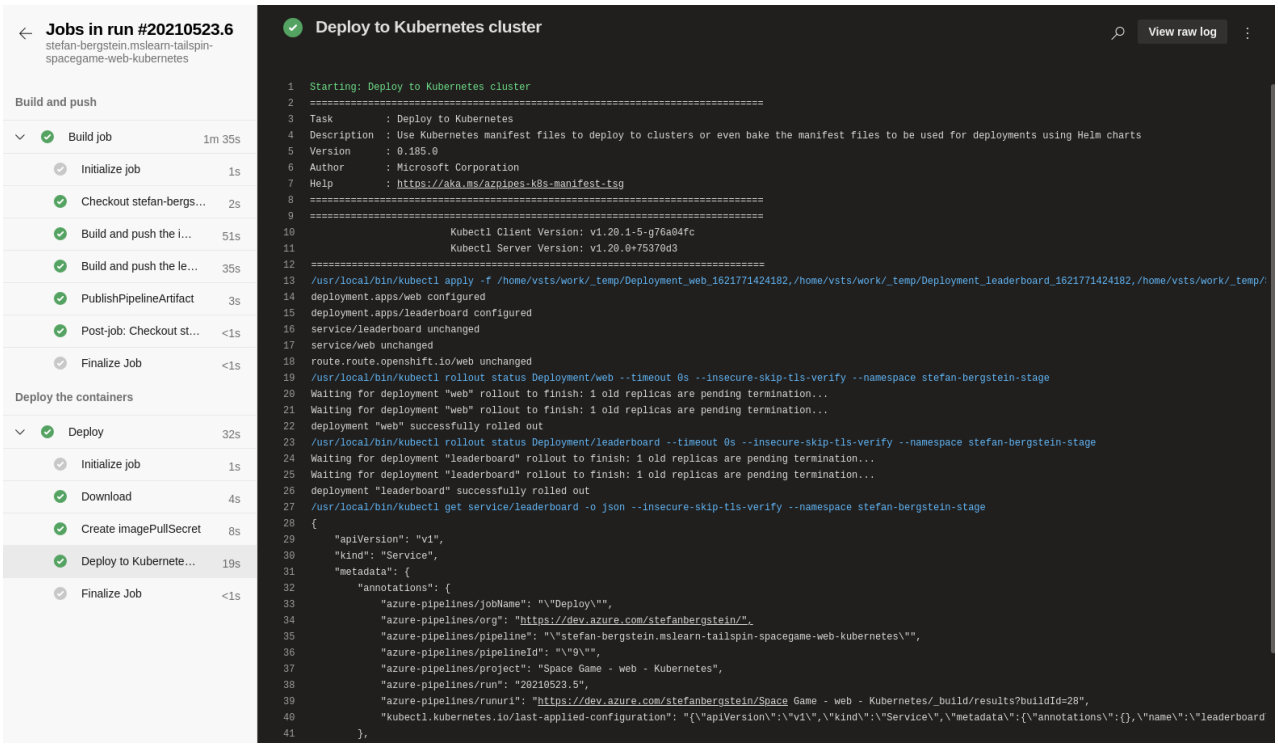


Figure 9.3 : Pipeline Azure DevOps poussant du contenu vers OpenShift

Pour en savoir plus

L'article suivant du blog de la communauté offre un excellent tutoriel sur les premiers pas avec Azure Red Hat OpenShift et Azure DevOps :

- Article de blog – Mai 2021

Intégration à Visual Studio Code

Les développeurs d'OpenShift peuvent apporter de la valeur ajoutée à l'intégration via les plug-ins dédiés à de nombreux IDE populaires, notamment Visual Studio Code. Les développeurs et les administrateurs peuvent ainsi accéder rapidement et facilement aux ressources Kubernetes et OpenShift sans devoir quitter leur IDE.

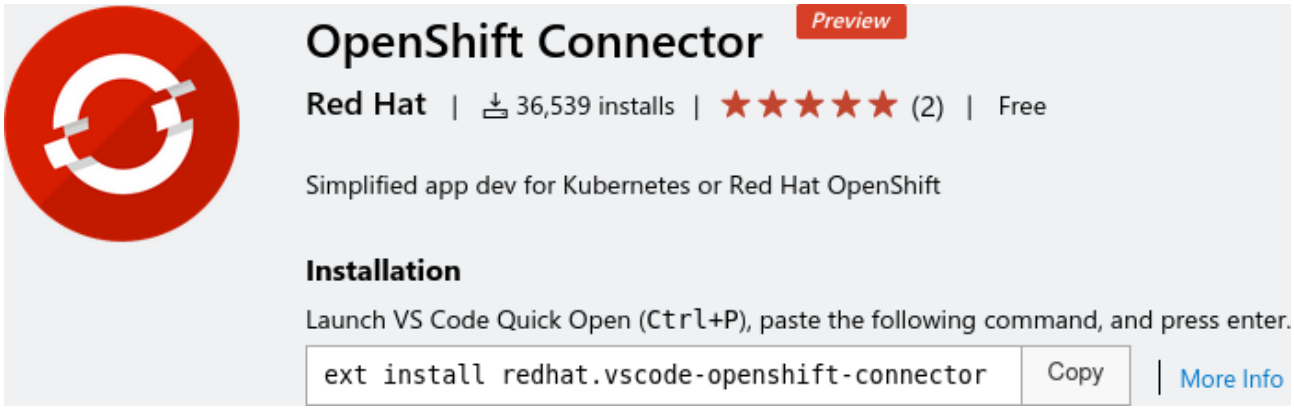


Figure 9.4 : Installation d'OpenShift Connector

Une fois le connecteur téléchargé et installé dans Visual Studio Code, vous serez invité à vous connecter à votre cluster OpenShift. Voici un projet simple de type « hello world » avec une connexion à Azure Red Hat OpenShift :

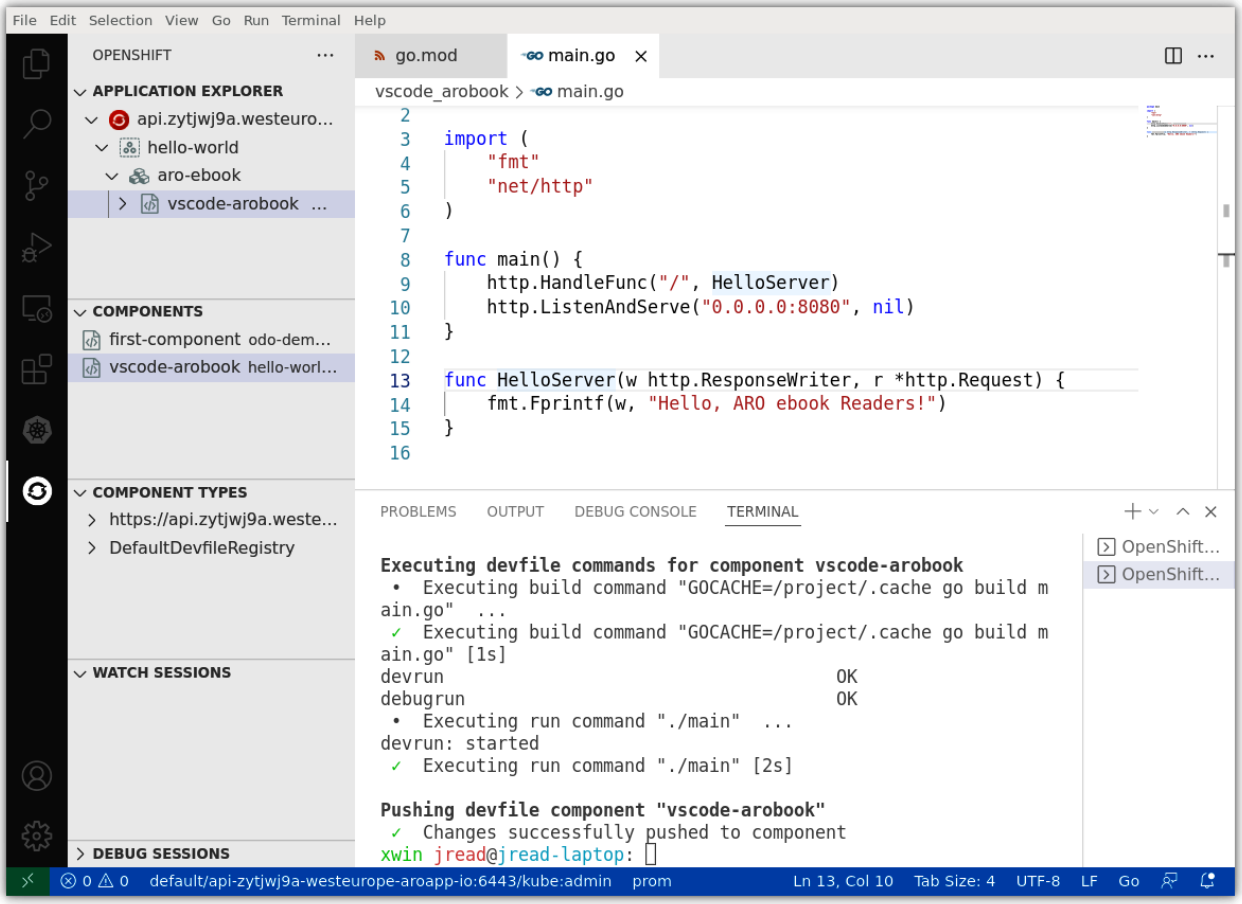


Figure 9.5 : Projet Golang qui vient d'être déployé avec le connecteur Visual Studio Code OpenShift

L'utilisation du connecteur OpenShift avec Visual Studio Code offre notamment l'avantage de la disponibilité d'un front-end graphique pour l'outil populaire OpenShift « OpenShift Do » (« odo »). Les développeurs peuvent ainsi réfléchir à des concepts de plus haut niveau que les simples composants Kubernetes bruts. Les développeurs ne doivent pas se préoccuper à ce point des déploiements, des ReplicaSets, etc. La capture d'écran précédente montre qu'il s'agit d'un projet simple, avec un service HTTP exposé.

De plus, le connecteur intègre une fonctionnalité permettant de pousser les modifications de code directement vers OpenShift, sans devoir nécessairement utiliser le contrôle de code source au préalable. Cela s'avère vraiment utile pour le développement rapide, car il n'est plus nécessaire de pousser vers le contrôle de code source à chaque fois, créer un nouveau conteneur et le déployer.

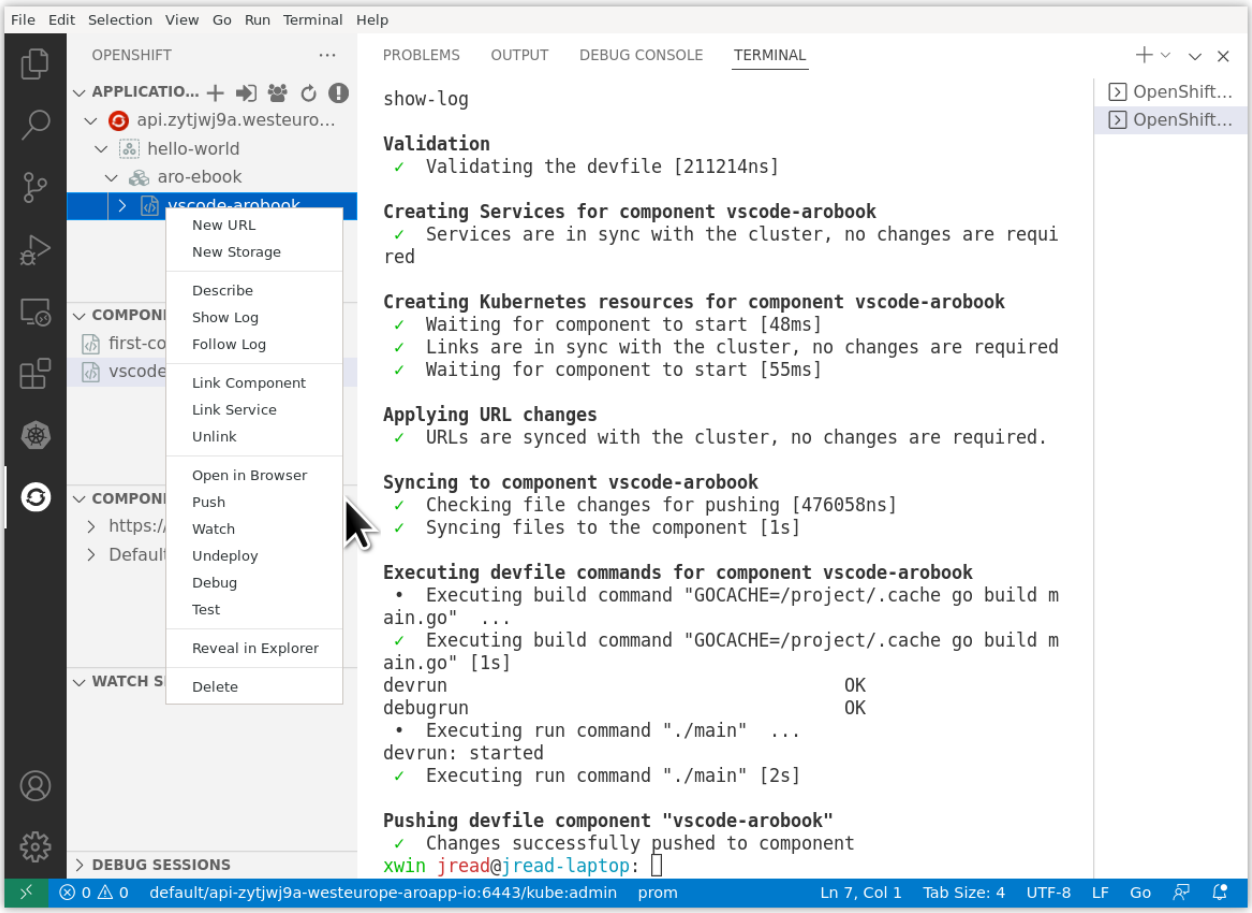


Figure 9.6 : Menu « push » (pousser) d'un projet et poussée récente dans la fenêtre de terminal

Ce connecteur accélère simplement les cycles de développement et de test des développeurs qui préfèrent utiliser Visual Studio Code.

Figure 9.7 : Application Golang de base fonctionnant sur OpenShift

Bien sûr, les développeurs ne sont pas limités à l'utilisation de Visual Studio Code. De nombreux développeurs utilisent Vim, Eclipse et d'autres éditeurs, mais Visual Studio Code est très populaire chez les développeurs qui apprécient une expérience de type « IDE léger ».

Pour en savoir plus

- Vidéo de démonstration – Utilisation de Visual Studio Code avec OpenShift
- Visual Studio Code OpenShift Connector

Intégration à GitHub Actions

Vous pouvez désormais utiliser GitHub Actions pour procéder à un déploiement vers n'importe quel environnement Red Hat OpenShift, y compris Azure Red Hat OpenShift. Depuis n'importe quel dépôt GitHub, accédez à **Actions** → **New Workflow** (Nouveau workflow) et sélectionnez **OpenShift** dans la liste des actions disponibles.

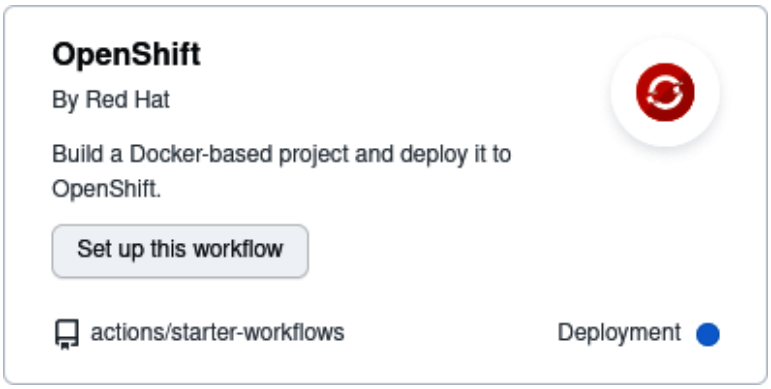


Figure 9.8 : Déploiement vers OpenShift à partir de GitHub

Le modèle par défaut est accompagné d'un excellent exemple d'ensemble d'actions qui ne nécessite que la définition de quelques variables d'environnement pour se connecter à un cluster OpenShift :

```
name: OpenShift

env:
  # 🖉 EDIT your repository secrets to log into your OpenShift cluster and set up the context.
  # See https://github.com/redhat-actions/oc-login#readme for how to retrieve these values.
  # To get a permanent token, refer to https://github.com/redhat-actions/oc-login/wiki/Using-a-
Service-Account-for-GitHub-Actions
  OPENSHIFT_SERVER: ${{ secrets.OPENSHIFT_SERVER }}
  OPENSHIFT_TOKEN: ${{ secrets.OPENSHIFT_TOKEN }}
  # 🖉 EDIT to set the kube context's namespace after login. Leave blank to use your user's
default namespace.
  OPENSHIFT_NAMESPACE: ""
  ...
```

Un excellent article de blog décrit comment procéder et une vidéo de démonstration est disponible :

- Article de blog
- GitHub Actions avec vidéo de démonstration sur OpenShift

Synthèse

Ce chapitre a abordé de nombreuses intégrations courantes que nos clients utilisent avec Azure Red Hat OpenShift. Comme indiqué dans l'introduction du chapitre, les API standard d'OpenShift permettent l'intégration de nombreux autres services non répertoriés ici. Vous pouvez également intégrer facilement un grand nombre de services grâce aux opérateurs répertoriés dans OperatorHub.

Dans le chapitre suivant, nous partagerons quelques bonnes pratiques et recommandations sur la manière d'intégrer les équipes chargées des applications et de tirer profit de l'adoption du service au sein de votre organisation.

Chapitre 10

Intégration des charges de travail et des équipes

Une fois que vous avez exécuté les étapes de *pré-provisionnement*, provisionné Azure Red Hat OpenShift et exécuté les étapes de post-provisionnement nécessaires, il ne reste plus qu'à prendre en charge et à embarquer les développeurs et les applications sur le cluster. La façon dont vous vous y prenez dépend surtout de la culture de votre organisation : certaines équipes de développeurs et d'applicateurs ont envie de « se jeter à l'eau » directement tandis que d'autres équipes apprécieront de recevoir davantage de conseils.

Ce chapitre regroupe un ensemble de listes de contrôle qui constituent une bonne base pour veiller à ce que vos développeurs et vos équipes en charge des applications soient opérationnels plus rapidement lorsqu'ils utilisent Azure Red Hat OpenShift.

Pensez à compléter ces listes de contrôle en les adaptant à votre structure organisationnelle et à votre culture.

Liste de contrôle : pré-intégration

Avant d'intégrer une nouvelle charge de travail, vous devez veiller à expliquer à l'équipe en charge des applications ou aux propriétaires de cette charge de travail qu'Azure Red Hat OpenShift a été déployé et conçu de manière à déjà correspondre aux meilleures pratiques de votre organisation :

- Azure Red Hat OpenShift a été déployé dans une zone d'atterrissage Azure ou un autre environnement Azure qui a été approuvé pour les applications de l'organisation.
- Le cluster dispose de toute la configuration du « Jour 2 » requise, comme les classes de stockage et l'authentification (décrite dans le *chapitre 6, Post-provisionnement – Jour 2*).
- Le cluster est déjà entièrement configuré et pris en charge par votre équipe en charge de la plateforme et bénéficie du support intégré offert par Red Hat et Microsoft.
- Le cluster a été mis à jour vers la dernière version stable disponible et les correctifs ont bien été appliqués. Les correctifs continueront d'être appliqués à l'avenir.

- Le cluster Azure Red Hat OpenShift dispose d'une connectivité aux ressources essentielles requises :
    - Connectivité à l'environnement sur site, via Azure ExpressRoute ou un VPN
    - Connectivité à un référentiel d'artefacts d'entreprise (tel qu'un référentiel Java Maven)
    - Connectivité à l'Internet public pour le téléchargement des dépendances pendant le développement d'une application

Intégration des charges de travail pilotes

Un modèle commun de déploiement d'Azure Red Hat OpenShift dans une organisation consiste à intégrer au moins deux charges de travail pilotes :

- **La charge de travail significative minimale** : idéalement, en tant qu'équipe SRE, la première charge de travail pilote est une petite application bien connue dont les exigences techniques sont très simples. Il s'agit d'une application légèrement supérieure à un simple « Hello World » dans la mesure où l'application doit normalement posséder un véritable propriétaire au sein de l'organisation. Cependant, avec la première charge de travail, l'accent est mis sur la vérification que le cluster Azure Red Hat OpenShift peut répondre aux besoins de l'entreprise au niveau du cluster (connectivité Azure, connexion Azure Active Directory, identification des tâches simples) en résolvant des problèmes communs que chaque application peut rencontrer.
- **Une charge de travail de référence significative** : après le déploiement d'une charge de travail simple et minimale, l'adoption est vraiment facilitée si la deuxième charge de travail est suffisamment complexe et significative (importante pour l'entreprise, voire critique pour l'entreprise). Cette charge de travail permettra d'identifier et de résoudre des problèmes tels que la connectivité aux bases de données importantes, les tests de performance et la journalisation/les mesures à grande échelle. Notez que cette charge de travail de référence significative peut ensuite être utilisée comme **référence interne** au sein de votre organisation. Les futures équipes de développement et d'application pourront plus facilement se familiariser avec la plateforme Azure Red Hat OpenShift.

D'autres modèles d'intégration des premières charges de travail pourraient bien entendu convenir à votre organisation. Vous devez cibler des applications dans un datacenter qui sera bientôt mis hors service ou cibler des applications qui fonctionnent sur un vieux serveur d'applications Java.

Liste de contrôle : réunion d'intégration avec les nouvelles équipes

Lors de l'intégration d'une nouvelle équipe de développement ou d'application sur le cluster, une bonne pratique consiste à organiser une réunion d'intégration :

- Créez un espace de noms ou une série d'espaces de noms que l'équipe pourra utiliser.
- Présentez Red Hat OpenShift à l'équipe au moyen d'une brève démonstration en soulignant les fonctionnalités clés telles que le déploiement à partir de GitHub (ou équivalent).
- Vérifiez que le cluster dispose d'une capacité de réserve suffisante pour déployer la charge de travail cible (processeur, RAM, stockage, etc.).
- Vérifiez que les membres de l'équipe peuvent se connecter au cluster. La connectivité à Azure Active Directory est un bon moyen de faciliter cette tâche.
- Fournissez les coordonnées de l'équipe SRE (ou similaire) qui gère le cluster.
- Fournissez une liste de liens permettant d'effectuer ses premiers pas dans OpenShift :

    - http://learn.openshift.com
    - https://github.com/openshift-labs/starter-guides
    - http://docs.openshift.com

Liste de contrôle : appels réguliers pour le contrôle d'intégrité

Une fois qu'une équipe de développement ou d'application a lancé le déploiement du cluster, organiser des réunions régulières de contrôle s'avère souvent utile. Ces activités pourraient faire partie d'une réunion quotidienne debout. Vous pourriez également n'y consacrer que 30 minutes par semaine. Voici une liste de points que vous pouvez vérifier :

- Comment se porte l'équipe en général : des applications ont-elles été déployées ?
- Des problèmes récents dans l'utilisation du cluster (connaissance de Red Hat OpenShift ou problèmes de connectivité) ont-ils été détectés ?
- Des pannes dans Azure ou avec le cluster avec une incidence négative sur l'expérience sont-elles survenues ?
- Rappel de la disponibilité de l'équipe SRE et de ses coordonnées pour répondre aux questions.

Réfléchissez aux autres points que vous avez précédemment vérifiés lors de contrôles d'intégrité réguliers pour des projets similaires. Ajoutez à cette liste les points qui, selon vous, pourraient être appropriés.

Anti-modèle : terrains de jeu et bacs à sable des développeurs

Pour encourager les développeurs à adopter Azure Red Hat OpenShift au sein d'une organisation, vous pouvez recourir à un « anti-modèle » commun. Fournissez un environnement de type bac à sable, communément appelé « terrain de jeu des développeurs », et attendez que les développeurs et les applications commencent à adopter Azure Red Hat OpenShift. Sans aucune assistance ou ressource de suivi supplémentaire, Azure Red Hat OpenShift peut constituer un environnement intimidant dont la grande complexité peut être trop importante à absorber. Dans la plupart des cas, l'adoption d'un modèle de type bac à sable peut entraîner un gaspillage des dépenses de cloud et des clusters vides.

Si toutefois votre organisation a l'habitude de mettre en place des bacs à sable avec les équipes de développement, voici quelques conseils pour optimiser les atouts de cette pratique :

- Fournissez au moins une brève démonstration des fonctionnalités d'Azure Red Hat OpenShift.
- Fournissez de la documentation et au moins les liens de démarrage décrits précédemment.
- Organisez un événement de type hackathon en mettant les développeurs au défi de créer quelque chose avec OpenShift dans le cadre d'une petite compétition.
- Proposez une offre d'assistance étendue, une présentation de l'équipe SRE et une expérience d'intégration mieux encadrée en option.

En suivant une liste de contrôle du modèle d'adoption guidée, comme indiqué précédemment, vous multipliez les chances que la plateforme soit adoptée.

Atelier pour développeurs Azure Red Hat OpenShift

L'atelier pour développeurs Azure Red Hat OpenShift (https://aroworkshop.io) est un atelier prédéfini utile que vous pouvez utiliser au sein de votre organisation pour offrir une expérience guidée et pratique d'Azure Red Hat OpenShift. L'atelier est divisé en deux exercices de laboratoire au sein d'un tutoriel conçu pour guider un développeur ou un opérateur novice dans l'utilisation d'OpenShift. Ces deux laboratoires présentent les principales fonctionnalités d'OpenShift et montrent comment elles peuvent être utilisées. Le laboratoire 1 est une introduction à la conception de microservices modernes. Le laboratoire 2 s'intéresse aux aspects internes du service.

Pour sensibiliser votre organisation à Azure Red Hat OpenShift, vous pouvez notamment utiliser le contenu d'aroworkshop.io sur votre propre cluster interne Azure Red Hat OpenShift. Lors d'une réunion préalable à l'atelier, vous pouvez expliquer comment Azure Red Hat OpenShift a été déployé dans le cadre de la souscription Azure Red Hat OpenShift existante de votre organisation, et que l'utilisation du service dans le cadre de l'atelier pourrait être une expérience similaire à l'utilisation d'Azure Red Hat OpenShift pour l'hébergement de leurs applications de production.

Synthèse

Ce chapitre a proposé des listes de contrôle et des conseils utiles pour réussir l'intégration des charges de travail et des équipes à Azure Red Hat OpenShift. Un anti-modèle commun a été décrit : des clusters de type bac à sable sans support. Un manuel peut difficilement présenter une liste complète de ressources et d'éléments de liste de contrôle qui s'appliqueraient à chaque organisation. C'est pourquoi vous devez adapter le contenu de ce chapitre à vos propres besoins.

Le cloud offre une gamme intéressante de services. Cependant, nombre d'entre eux sont négligés ou mal adoptés, car les organisations se concentrent essentiellement sur la technologie et la manière dont elle est déployée. Si la compréhension de ces thèmes est importante, cela ne constitue qu'une petite partie de l'équation lors du passage de ce service à la production et de son adoption efficace. Ce chapitre a tenté de mettre en évidence la manière dont la formation, les contrôles réguliers et les activités similaires sont nécessaires pour réussir votre adoption d'Azure Red Hat OpenShift.

Chapitre 11

Conclusion

Vos équipes de développement et d'exploitation peuvent consacrer la majeure partie de leur temps de travail à s'occuper du provisionnement, de la configuration, de la maintenance et de la supervision de vos clusters et du pipeline CI/CD. Dans ce cas, ils ne peuvent pas consacrer leur temps précieux à ce qu'ils font le mieux : assurer le bon fonctionnement de vos applications à la pointe du progrès.

Comme nous l'avons appris dans ce manuel, Azure Red Hat OpenShift permet de déployer des clusters Red Hat OpenShift entièrement gérés sans se soucier de la création ou de la gestion de l'infrastructure nécessaire à leur fonctionnement. Nous avons vu que l'exécution de Kubernetes seul s'accompagne de quelques réserves, principalement en ce qui concerne l'attention manuelle supplémentaire requise pour des tâches qui pourraient être automatisées avec Azure Red Hat OpenShift.

Lorsque vous décidez de la stratégie de gestion des clusters de votre organisation, tenez compte des avantages et des inconvénients d'une plateforme de type Kubernetes par rapport à Azure Red Hat OpenShift, qui est conçu sur le framework Kubernetes et vous offre un ensemble d'avantages supplémentaires prêts à l'emploi.

Pour en savoir plus sur Azure Red Hat OpenShift, visitez la page du produit ou consultez notre section de documentation. Vous pouvez également participer à un atelier pratique et vous inscrire pour regarder un webinaire à votre convenance. Nous espérons surtout que vous vous adresserez à Microsoft et à Red Hat pour vous aider à évaluer Azure Red Hat OpenShift.

Auteurs et versions

James Read <james@redhat.com>
Architecte principal de solutions chez Red Hat,
pour Microsoft – mise à jour et révision pour AROv4

Ahmed Sabbour <asabbour@microsoft.com>
Responsable principal du marketing produit chez Microsoft,
pour Azure Red Hat OpenShift – version initiale d'AROv3

Auteurs des éditions précédentes

Oren Kashi <okashi@redhat.com>
Responsable principal du marketing technique des produits chez Red Hat

Nous tenons à remercier Brooke Jackson, Nermina Miller, Jose Moreno, Ahmed Sabbour, Aditya Datar, Vince Power, Alex Patterson et les autres personnes qui ont aimablement offert leur temps et leurs commentaires pour la révision de ce manuel.

Chapitre 12

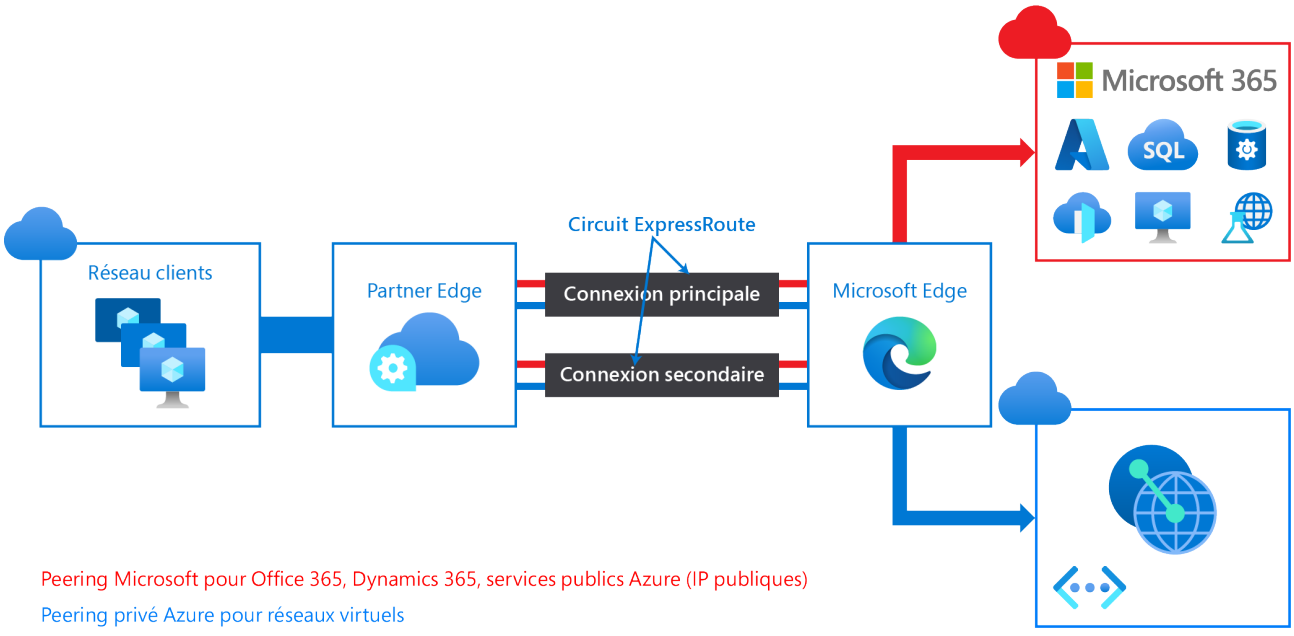
Glossaire

Ce glossaire constitue une référence rapide et utile de certains termes utilisés dans ce manuel et dans l'écosystème Azure Red Hat OpenShift. Les termes sont classés par ordre alphabétique.

Azure ExpressRoute

ExpressRoute vous permet d'étendre vos réseaux sur site au cloud Microsoft via une connexion privée avec l'aide d'un fournisseur de connectivité. Avec ExpressRoute, vous pouvez établir des connexions aux services cloud de Microsoft, tels que Microsoft Azure et Microsoft 365.

Voici un schéma de réseau ExpressRoute issu de la documentation de Microsoft Azure :



Pour en savoir plus sur ExpressRoute, consultez la page Qu'est-ce qu'Azure ExpressRoute ? de la documentation Microsoft Azure.

Pour comprendre comment ExpressRoute fonctionne avec Azure, consultez le *chapitre 4, Pré-provisionnement – Questions sur l'architecture d'entreprise*.

Builds et flux d'images

Une build est le processus de transformation des paramètres d'entrée en un objet résultant. La plupart du temps, le processus sert à transformer les paramètres d'entrée ou le code source en une image exécutable. Un objet BuildConfig est la définition de l'ensemble du processus de création de build.

Azure Red Hat OpenShift utilise Kubernetes en créant des conteneurs au format Docker à partir d'images de build et en les transférant vers un registre d'images de conteneurs.

Les objets de build ont en commun certaines caractéristiques : les entrées pour une build, la nécessité d'effectuer en entier un processus de création de build, la journalisation du processus de création de build, la publication de ressources provenant de builds finalisées et la publication du statut final de la build. Les builds tirent parti des restrictions de ressources, en spécifiant les limitations sur les ressources comme l'utilisation du processeur, l'utilisation de la mémoire et le temps d'exécution de build ou de pod.

Le système de création de build d'Azure Red Hat OpenShift offre une prise en charge extensible des stratégies de build qui s'appuient sur des types sélectionnables spécifiés dans l'API de build. Il existe trois stratégies principales de création de build :

- **Build Docker** : avec un fichier Docker, qui peut être téléchargé ou tiré d'un dépôt de sources
- **Build Source-to-Image (S2I)** : prend un dépôt de sources (tel que Git) et détermine comment créer l'application à partir de fichiers de création en langage connu (par exemple, les fichiers Maven `.pom` pour les projets Java)
- **Build personnalisée**

Par défaut, les builds Docker et les builds S2I sont prises en charge.

L'objet résultant d'une build dépend de l'outil de création utilisé. Dans le cas des builds Docker et S2I, les objets résultant sont des images exécutables. Dans celui des builds personnalisées, les objets résultant sont liés aux éléments spécifiés par l'auteur de l'image de l'outil de création.

Conteneur

Les unités de base des applications Azure Red Hat OpenShift sont appelées des conteneurs. Les technologies de conteneurs Linux sont des mécanismes légers pour isoler les processus en cours d'exécution afin qu'ils n'interagissent qu'avec leurs ressources désignées.

Il est possible d'exécuter de nombreuses instances d'application dans des conteneurs sur un seul hôte sans qu'il n'existe de visibilité mutuelle entre les processus, les fichiers, le réseau, etc. Généralement, chaque conteneur fournit un service unique (souvent appelé « microservice »), tel qu'un serveur web ou une base de données, bien que les conteneurs puissent être utilisés pour des charges de travail arbitraires.

Déploiements et configurations de déploiement

Reposant sur des contrôleurs de réplication, Azure Red Hat OpenShift ajoute la prise en charge étendue pour le cycle de vie du développement et du déploiement de logiciels avec le concept de déploiements. Dans le cas le plus simple, un déploiement crée simplement un nouveau contrôleur de réplication et le laisse démarrer les pods. Toutefois, les déploiements Azure Red Hat OpenShift offrent également la possibilité de passer d'un déploiement existant d'une image à un nouveau déploiement. Ils définissent aussi les scripts automatiques à exécuter avant ou après la création du contrôleur de réplication.

L'objet Azure Red Hat OpenShift DeploymentConfig définit les détails suivants d'un déploiement :

1. Les éléments d'une définition ReplicationController.
2. Les déclencheurs pour la création automatique d'un déploiement.
3. La stratégie de transition entre les déploiements.
4. Les scripts automatiques de cycle de vie.

Chaque fois qu'un déploiement est déclenché, que ce soit manuellement ou automatiquement, un pod de déploiement gère le déploiement (y compris la réduction de l'ancien contrôleur de réplication, la mise à l'échelle du nouveau et l'exécution des scripts automatiques). Après la fin du déploiement, le pod de déploiement continue d'exister pendant une durée indéfinie afin que ses journaux de déploiement soient conservés. Lorsqu'un déploiement est remplacé par un autre, le contrôleur de réplication précédent est conservé afin de permettre une restauration facile si nécessaire.

Pour obtenir des instructions détaillées sur la création et l'interaction avec les déploiements, reportez-vous à Déploiements et DeploymentConfigs.

Ensemble de réplicas

Semblable à un contrôleur de réplication, un ensemble de réplicas garantit qu'un nombre spécifique de réplicas de pods est exécuté à tout moment. La différence entre un ensemble de réplicas et un contrôleur de réplication repose sur le fait qu'un ensemble de réplicas prend en charge les exigences de sélecteur basées sur un ensemble alors qu'un contrôleur de réplication ne prend en charge que les exigences de sélecteur basées sur l'égalité.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
    name: frontend-1
    labels:
        tier: frontend
spec:
    replicas: 3
    selector:
        matchLabels:
            tier: frontend
        matchExpressions:
        - {key: tier, operator: In, values: [frontend]}
    template:
        metadata:
            labels:
                tier: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name : helloworld
                  ports:
                  - containerPort: 8080
                    protocol: TCP
            restartPolicy: Always
```

Le code ci-dessus contient les éléments suivants :

- Une requête d'étiquette sur un ensemble de ressources. Les résultats de `matchLabels` et `matchExpressions`sont logiquement conjoints.
- Un sélecteur basé sur l'égalité pour spécifier les ressources dont les étiquettes correspondent au sélecteur.
- Un sélecteur basé sur un ensemble pour filtrer les clés. Toutes les ressources sont sélectionnées avec une `clé` égale à `tier` et une `valeur` égale à `frontend`.

Images de conteneur

Les conteneurs dans Azure Red Hat OpenShift sont basés sur des images de conteneurs au format Docker. Une image est un fichier binaire qui inclut toutes les exigences pour exécuter un seul conteneur, ainsi que des métadonnées décrivant ses besoins et ses fonctionnalités.

Vous pouvez l'imaginer comme une technologie de mise en paquets. Les conteneurs n'ont accès qu'aux ressources définies dans l'image, à moins que vous n'attribuiez un accès supplémentaire au conteneur lors de sa création. En déployant la même image dans plusieurs conteneurs sur plusieurs hôtes et en équilibrant la charge entre eux, Azure Red Hat OpenShift peut fournir la redondance et la mise à l'échelle horizontale pour un service intégré dans une image.

Modèles

Un modèle décrit un ensemble d'objets qui peuvent être paramétrés et traités afin de produire une liste d'objets qui seront créés par Azure Red Hat OpenShift. Il peut être traité pour créer tout ce qu'il est autorisé de créer dans un projet, par exemple des services, de configurations de build et de configurations de déploiement. Un modèle peut également définir un ensemble d'étiquettes à appliquer à chaque objet défini dans le modèle.

Vous pouvez créer une liste d'objets à partir d'un modèle à l'aide de l'interface en ligne de commande ou, si un modèle a été téléchargé dans votre projet ou dans la bibliothèque de modèles globale, à l'aide de la console web. Pour un ensemble de modèles sélectionnés, consultez la bibliothèque OpenShift Image Streams and Templates.

Pods et services

Azure Red Hat OpenShift utilise le concept de Kubernetes d'un pod. Il est représenté par un ou plusieurs conteneurs déployés ensemble sur un hôte. Il s'agit de la plus petite unité de calcul pouvant être définie, déployée et gérée.

Les pods sont à peu près l'équivalent d'une instance de machine (physique ou virtuelle) pour un conteneur. Chaque pod se voit attribuer sa propre adresse IP interne et devient ainsi propriétaire de tout son espace de port. Les conteneurs dans les pods peuvent partager leur stockage local et leur mise en réseau.

Les pods ont un cycle de vie : ils sont définis et affectés pour s'exécuter sur un nœud. Ils fonctionnent ensuite jusqu'à ce que leurs conteneurs sortent ou qu'ils soient supprimés pour une autre raison. Selon la politique et le code de sortie, vous pouvez supprimer les pods après la sortie ou les conserver afin de permettre l'accès aux journaux de leurs conteneurs.

Azure Red Hat OpenShift traite les pods comme étant essentiellement immuables ; une définition de pod ne peut être modifiée pendant l'exécution de ce dernier. Azure Red Hat OpenShift met en œuvre les modifications en mettant fin à un pod existant et en le recréant avec une configuration modifiée, des images de base, ou les deux. Les composants d'exécution d'un pod sont considérés comme des consommables et sont recréés à partir de l'image de conteneur définie. Par conséquent, les pods doivent généralement être gérés par des contrôleurs de plus haut niveau, plutôt que directement par les utilisateurs.

Projets et utilisateurs

Un projet est un espace de noms Kubernetes comprenant des annotations supplémentaires. Il est également le moyen central de gestion de l'accès des utilisateurs réguliers aux ressources. Un projet permet à une communauté d'utilisateurs d'organiser et de gérer du contenu indépendamment des autres communautés. Les administrateurs fournissent aux utilisateurs des autorisations d'accès aux projets. Toutefois, si les utilisateurs sont autorisés à créer des projets, ils ont automatiquement accès à leurs propres projets. Les projets peuvent comporter un nom, un `displayName` et une description distincts.

Le nom obligatoire est un identifiant unique pour le projet et est l'élément le plus visible lors de l'utilisation des outils CLI (interface en ligne de commande) ou de l'API. La longueur maximale du nom est de 63 caractères. Le `displayName` facultatif représente la façon dont le projet est affiché dans la console web (le `nom` par défaut). La description facultative peut être une description plus détaillée du projet et est également visible dans la console web.

Les développeurs et les administrateurs peuvent interagir avec les projets à l'aide de l'interface en ligne de commande ou la console web.

Registre de conteneurs

Azure Red Hat OpenShift fournit un registre d'images de conteneur intégré appelé **OpenShift Container Registry (OCR)** qui permet de provisionner automatiquement de nouveaux référentiels d'images à la demande. Les utilisateurs disposent ainsi d'un emplacement intégré où leurs builds d'application envoient les images obtenues.

Chaque fois qu'une nouvelle image est envoyée à OCR, le registre le notifie à Azure Red Hat OpenShift, en transmettant toutes les informations la concernant, comme l'espace de noms, le nom et les métadonnées de l'image. Différents éléments d'Azure Red Hat OpenShift réagissent aux nouvelles images, en créant des builds et des déploiements.

Azure Red Hat OpenShift peut également utiliser n'importe quel serveur mettant en œuvre l'API de registre d'images de conteneurs en tant que source d'images, notamment Docker Hub et Azure Container Registry.

ReplicationController

Un contrôleur de réplication (ReplicationController) garantit qu'un nombre spécifique de réplicas est exécuté à tout moment. Si les pods disparaissent ou sont supprimés, le contrôleur de réplication agit pour en instancier d'autres, jusqu'au nombre défini. De même, s'il existe plus de pods en cours d'exécution, il en supprime autant qu'il faut pour respecter le nombre défini.

Une configuration de contrôleur de réplication comprend :

- le nombre de réplicas souhaité (qui peut être ajusté au moment de l'exécution) ;
- une définition de pod à utiliser lors de la création d'un pod répliqué ;
- un sélecteur pour identifier les pods gérés ;
- un sélecteur est un ensemble d'étiquettes attribuées aux pods dont la gestion est assurée par le contrôleur de réplication. Ces étiquettes sont incluses dans la définition de pod qui est instanciée par le contrôleur de réplication. Le contrôleur de réplication utilise le sélecteur pour déterminer le nombre d'instances du pod déjà en cours d'exécution afin de procéder à des ajustements si nécessaire.

Le contrôleur de réplication n'effectue pas de mise à l'échelle automatique en fonction de la charge ou du trafic, car il n'effectue aucun suivi. Cela impliquerait que son nombre de réplicas soit ajusté par une mise à l'échelle automatique externe.

Un contrôleur de réplication est un objet Kubernetes de base appelé ReplicationController. Voici un exemple de définition de ReplicationController :

```
apiVersion: v1
kind: ReplicationController
metadata:
    name: frontend-1
spec:
    replicas: 1
    selector:
        name: frontend
    template:
        metadata:
            labels:
                name: frontend
        spec:
            containers:
                - image: openshift/hello-openshift
                  name: helloworld
                  ports:
                - containerPort: 8080
                  protocol: TCP
            restartPolicy: Always
```

Le code ci-dessus contient les éléments suivants :

- Le nombre de copies du pod à exécuter
- Le sélecteur d'étiquette du pod à exécuter
- Un modèle pour le pod que le contrôleur crée
- Les étiquettes du pod doivent inclure celles du sélecteur d'étiquettes
- La longueur maximale du nom après développement de paramètres est de 63 caractères

Routes et entrées

Azure Red Hat OpenShift prend en charge les routes et les entrées. Ils sont utilisés pour exposer un service à l'aide d'un nom DNS, tel que www.example.com, afin que des clients externes puissent l'atteindre.

Les routes ont été conçues à l'origine dans la version 3 de Red Hat OpenShift. Depuis sa sortie, Red Hat a travaillé avec la communauté Kubernetes pour standardiser cette fonctionnalité qui s'appelle maintenant **Ingress** (ou entrée).

La ressource Kubernetes Ingress dans OpenShift Container Platform met en œuvre le contrôleur Ingress avec un service de routeur partagé qui s'exécute en tant que pod dans le cluster. La façon la plus courante de gérer le trafic d'entrée est le contrôleur d'entrée. Vous pouvez adapter et reproduire ce pod comme n'importe quel autre pod ordinaire. Ce service de routeur repose sur HAProxy, une solution open-source d'équilibrage de charge.

La route OpenShift Container Platform fournit le trafic d'entrée aux services du cluster. Les routes offrent des fonctionnalités avancées qui pourraient ne pas être prises en charge par les contrôleurs d'entrée Kubernetes standard, telles que le rechiffrement TLS, le passage TLS et la division du trafic pour les déploiements bleu-vert.

Le trafic d'entrée accède aux services du cluster par le biais d'une route. Les routes et les entrées sont les principales ressources qui permettent de traiter le trafic d'entrée. Une entrée offre des fonctionnalités similaires à celles d'une route, comme l'acceptation de demandes externes et leur délégation en fonction de la route. Cependant, une entrée ne vous permet d'autoriser que certains types de connexions : HTTP/2, HTTPS et **identification du nom du serveur (SNI)**, et TLS avec un certificat. Dans OpenShift Container Platform, les routes sont générées pour répondre aux conditions spécifiées par la ressource Ingress.

Source-to-Image (S2I)

S2I est une boîte à outils et un workflow pour la création d'images de conteneurs reproductibles et prêtes à l'emploi à partir du code source. S2I fournit des images prêtes à l'emploi. Pour ce faire, le code source est injecté dans une image de conteneur qui se charge de le préparer pour l'exécution. En créant des images d'outils de création assemblées automatiquement, vous pouvez créer et contrôler les versions de vos environnements de build tout comme vous utilisez des images de conteneurs pour créer des versions de vos environnements d'exécution.

Dans le cas d'un langage dynamique comme Ruby, les environnements au moment de la création et les environnements au moment de l'exécution sont traditionnellement les mêmes. En prenant comme point de départ une image d'outil de création qui décrit cet environnement (avec Ruby, Bundler, Rake, Apache, GCC et d'autres packages nécessaires pour configurer et exécuter une application Ruby installée), S2I effectue les étapes suivantes :

1. Démarrage d'un conteneur à partir de l'image d'outil de création avec la source de l'application injectée dans un répertoire connu.
2. Le processus de conteneur transforme ce code source en configuration exécutable ; dans ce cas, en installant des dépendances avec Bundler et en déplaçant le code source dans un répertoire où Apache a été préconfiguré pour rechercher le fichier `config.ru` de Ruby.
3. Validation du nouveau conteneur et définition du point d'entrée de l'image en tant que script (fourni par l'image d'outil de création) qui lancera Apache pour héberger l'application Ruby.

Pour les langages compilés comme C, Go ou Java, les dépendances nécessaires pour la compilation peuvent considérablement dépasser la taille des artefacts d'exécution réels. Pour garder des images d'exécution de taille raisonnable, S2I offre un processus de création de build comprenant plusieurs étapes. Un artefact binaire tel qu'un fichier exécutable ou Java WAR est créé dans la première image d'outil de création, extrait et injecté dans une deuxième image d'exécution qui place simplement le fichier exécutable au bon endroit pour l'exécution.

Par exemple, pour créer un pipeline de build reproductible pour Tomcat (serveur web Java connu) et Maven , procédez comme suit :

1. Créez une image d'outils de création contenant OpenJDK et Tomcat dans laquelle un fichier WAR va être injecté.
2. Créez une deuxième image qui se superpose à la première image Maven et aux autres dépendances standard, et dans laquelle un projet Maven va être injecté.
3. Appelez S2I en utilisant la source de l'application Java et l'image Maven pour créer l'application WAR souhaitée.
4. Appelez à nouveau S2I en utilisant le fichier WAR de l'étape précédente et l'image Tomcat initiale pour créer l'image d'exécution.

En plaçant notre logique de build à l'intérieur des images et en associant les images en plusieurs étapes, nous conservons des environnements d'exécution et de création de build (JDK similaire, JAR Tomcat similaires) relativement similaires sans devoir déployer des outils de création de build en production.

L'utilisation de S2I comme stratégie de build présente les objectifs et les atouts suivants :

- **Reproductibilité** : la version des environnements de build est étroitement contrôlée ; ils sont encapsulés dans une image de conteneur tandis qu'une interface simple (code source injecté) est définie pour les appelants. Le caractère reproductible des builds est une exigence clé, car il permet des mises à jour de sécurité et une intégration continue dans une infrastructure conteneurisée. En outre, les images d'outils de création garantissent ce caractère reproductible ainsi que la possibilité de permuter des environnements d'exécution.
- **Flexibilité** : tout système de build existant pouvant s'exécuter sous Linux peut l'être à l'intérieur d'un conteneur, et chaque outil de création de build individuel peut également faire partie d'un pipeline plus important. De plus, les scripts qui traitent le code source de l'application peuvent être injectés dans l'image d'outil de création. Les auteurs peuvent ainsi adapter les images existantes pour permettre le traitement des sources.
- **Rapidité** : au lieu de créer plusieurs couches dans un seul fichier Docker, S2I encourage les auteurs à représenter une application dans une seule couche d'image. Cela engendre un gain de temps lors de la création et du déploiement et permet un meilleur contrôle sur la sortie de l'image finale.

- **Sécurité** : les builds utilisant des fichiers Docker sont exécutées sans la plupart des contrôles opérationnels classiques de conteneurs. Elles sont généralement exécutées en tant que root et ont accès au réseau de conteneurs. S2I peut être utilisé pour contrôler les autorisations et les privilèges disponibles pour l'image d'outil de création dans la mesure où la build est lancée dans un seul conteneur. De concert avec des plateformes comme OpenShift, S2I peut permettre aux administrateurs de contrôler étroitement les privilèges dont disposent les développeurs au moment de la création de build.

Tâches

Une tâche est similaire à un contrôleur de réplication dans la mesure où son objectif est de créer des pods pour des raisons précises. La différence repose sur le fait que les contrôleurs de réplication sont conçus pour les pods qui s'exécuteront en continu, tandis que les tâches sont destinées aux pods ponctuels. Une tâche effectue le suivi de toutes les réalisations achevées et lorsque le nombre spécifié de réalisations a été atteint, la tâche elle-même est terminée.

Consultez la rubrique *Tâches* pour obtenir plus d'informations sur la façon d'utiliser les tâches.

Zones d'atterrissage Azure

Les zones d'atterrissage Azure sont un modèle de déploiement populaire pour les organisations qui planifient un déploiement à grande échelle à l'aide d'Azure en tenant compte de l'échelle, de la gouvernance de la sécurité, de la mise en réseau et de l'identité.

[Présentation des zones d'atterrissage Azure]

Un projet communautaire GitHub, actuellement en cours de développement au moment de la rédaction de ce manuel, formule des recommandations sur la manière de déployer Azure Red Hat OpenShift dans une architecture de zone d'atterrissage Azure : https://github.com/Azure/Enterprise-Scale/tree/main/workloads/ARO