Plate-forme de conteneurs OpenShift 4.12

L'architecture

Vue d'ensemble de l'architecture d'OpenShift Container Platform

Last Updated: 2023-04-26

Plate-forme de conteneurs OpenShift 4.12 L'architecture

Vue d'ensemble de l'architecture d'OpenShift Container Platform

Notice légale

Résumé

Ce document fournit une vue d'ensemble de l'architecture de la plateforme et des applications d'OpenShift Container Platform.

Table des matières

CHAPITRE 1. APERÇU DE L'ARCHITECTURE **4**
1.1. GLOSSAIRE DES TERMES COURANTS RELATIFS À L'ARCHITECTURE D'OPENSHIFT CONTAINER PLATFORM 4
1.2. A PROPOS DE L'INSTALLATION ET DES MISES À JOUR 8
1.3. À PROPOS DU PLAN DE CONTRÔLE 8
1.4. A PROPOS DES APPLICATIONS CONTENEURISÉES POUR LES DÉVELOPPEURS 9
1.5. À PROPOS DE RED HAT ENTERPRISE LINUX COREOS (RHCOS) ET IGNITION 9
1.6. A PROPOS DES PLUGINS D'ADMISSION 10

CHAPITRE 2. ARCHITECTURE DE LA PLATEFORME DE CONTENEURS OPENSHIFT **11**
2.1. INTRODUCTION À OPENSHIFT CONTAINER PLATFORM 11

CHAPITRE 3. INSTALLATION ET MISE À JOUR **17**
3.1. A PROPOS DE L'INSTALLATION D'OPENSHIFT CONTAINER PLATFORM 17
3.2. À PROPOS DU SERVICE DE MISE À JOUR OPENSHIFT 25
3.3. POLITIQUE DE SOUTIEN AUX OPÉRATEURS NON GÉRÉS 26
3.4. PROCHAINES ÉTAPES 27

CHAPITRE 4. RED HAT OPENSHIFT CLUSTER MANAGER **28**
4.1. ACCÈS À RED HAT OPENSHIFT CLUSTER MANAGER 28
4.2. ACTIONS GÉNÉRALES 28
4.3. ONGLETS DE LA GRAPPE 29
4.4. RESSOURCES COMPLÉMENTAIRES 31

CHAPITRE 5. À PROPOS DU MOTEUR MULTICLUSTER POUR L'OPÉRATEUR KUBERNETES **32**
5.1. GESTION DES CLUSTERS AVEC LE MOTEUR MULTICLUSTER SUR OPENSHIFT CONTAINER PLATFORM 32
5.2. GESTION DE CLUSTERS AVEC RED HAT ADVANCED CLUSTER MANAGEMENT 32
5.3. RESSOURCES COMPLÉMENTAIRES 32

CHAPITRE 6. ARCHITECTURE DU PLAN DE CONTRÔLE **34**
6.1. GESTION DE LA CONFIGURATION DES NŒUDS AVEC DES POOLS DE CONFIGURATION DE MACHINES 34
6.2. RÔLES DES MACHINES DANS OPENSHIFT CONTAINER PLATFORM 35
6.3. OPÉRATEURS DANS OPENSHIFT CONTAINER PLATFORM 39
6.4. À PROPOS DE L'OPÉRATEUR DE CONFIGURATION DE LA MACHINE 41
6.5. APERÇU DES PLANS DE CONTRÔLE HÉBERGÉS (APERÇU TECHNOLOGIQUE) 43

CHAPITRE 7. COMPRENDRE LE DÉVELOPPEMENT DE LA PLATEFORME DE CONTENEURS OPENSHIFT .. **46**
7.1. A PROPOS DU DÉVELOPPEMENT D'APPLICATIONS CONTENEURISÉES 46
7.2. CONSTRUCTION D'UN CONTENEUR SIMPLE 46
7.3. CRÉER UN MANIFESTE KUBERNETES POUR OPENSHIFT CONTAINER PLATFORM 50
7.4. DÉVELOPPER POUR LES OPÉRATEURS 52

CHAPITRE 8. RED HAT ENTERPRISE LINUX COREOS (RHCOS) **54**
8.1. À PROPOS DU RHCOS 54
8.2. VISUALISATION DES FICHIERS DE CONFIGURATION D'IGNITION 59
8.3. MODIFICATION DE LA CONFIGURATION DE L'ALLUMAGE APRÈS L'INSTALLATION 62

CHAPITRE 9. PLUGINS D'ADMISSION **64**
9.1. A PROPOS DES PLUGINS D'ADMISSION 64
9.2. PLUGINS D'ADMISSION PAR DÉFAUT 64
9.3. PLUGINS D'ADMISSION AUX WEBHOOKS 67

9.4. TYPES DE PLUGINS D'ADMISSION AUX WEBHOOKS 68
9.5. CONFIGURATION DE L'ADMISSION DYNAMIQUE 71
9.6. RESSOURCES COMPLÉMENTAIRES 78

CHAPITRE 1. APERÇU DE L'ARCHITECTURE

OpenShift Container Platform est une plateforme de conteneurs Kubernetes basée sur le cloud. La fondation d'OpenShift Container Platform est basée sur Kubernetes et partage donc la même technologie. Pour en savoir plus sur OpenShift Container Platform et Kubernetes, voir l'architecture du produit.

1.1. GLOSSAIRE DES TERMES COURANTS RELATIFS À L'ARCHITECTURE D'OPENSHIFT CONTAINER PLATFORM

Ce glossaire définit les termes communs utilisés dans le contenu de l'architecture.

politiques d'accès

Ensemble de rôles qui déterminent la manière dont les utilisateurs, les applications et les entités d'une grappe interagissent les uns avec les autres. Une politique d'accès renforce la sécurité de la grappe.

plugins d'admission

Les plugins d'admission appliquent des politiques de sécurité, des limitations de ressources ou des exigences de configuration.

l'authentification

Pour contrôler l'accès à un cluster OpenShift Container Platform, un administrateur de cluster peut configurer l'authentification des utilisateurs et s'assurer que seuls les utilisateurs approuvés accèdent au cluster. Pour interagir avec un cluster OpenShift Container Platform, vous devez vous authentifier auprès de l'API OpenShift Container Platform. Vous pouvez vous authentifier en fournissant un jeton d'accès OAuth ou un certificat client X.509 dans vos demandes à l'API OpenShift Container Platform.

amorçage

Une machine temporaire qui exécute Kubernetes minimal et déploie le plan de contrôle d'OpenShift Container Platform.

les demandes de signature de certificat (CSR)

Une ressource demande à un signataire désigné de signer un certificat. Cette demande peut être approuvée ou refusée.

Opérateur de version de cluster (CVO)

Un opérateur qui vérifie avec le service de mise à jour d'OpenShift Container Platform les mises à jour valides et les chemins de mise à jour basés sur les versions actuelles des composants et les informations dans le graphe.

nœuds de calcul

Nœuds chargés d'exécuter les charges de travail pour les utilisateurs de la grappe. Les nœuds de calcul sont également appelés nœuds de travail.

dérive de la configuration

Situation dans laquelle la configuration d'un nœud ne correspond pas à ce que la configuration de la machine spécifie.

conteneurs

Il s'agit d'images légères et exécutables composées d'un logiciel et de toutes ses dépendances. Comme les conteneurs virtualisent le système d'exploitation, vous pouvez les exécuter n'importe où, que ce soit dans un centre de données, un nuage public ou privé ou votre hôte local.

moteur d'orchestration de conteneurs

Logiciel qui automatise le déploiement, la gestion, la mise à l'échelle et la mise en réseau des conteneurs.

charges de travail en conteneur

Applications conditionnées et déployées dans des conteneurs.

groupes de contrôle (cgroups)

Répartit les ensembles de processus en groupes afin de gérer et de limiter les ressources consommées par les processus.

plan de contrôle

Une couche d'orchestration de conteneurs qui expose l'API et les interfaces pour définir, déployer et gérer le cycle de vie des conteneurs. Les plans de contrôle sont également connus sous le nom de machines de plan de contrôle.

CRI-O

Une implémentation native de l'exécution des conteneurs Kubernetes qui s'intègre au système d'exploitation pour offrir une expérience Kubernetes efficace.

déploiement

Un objet de ressource Kubernetes qui maintient le cycle de vie d'une application.

Fichier Docker

Fichier texte contenant les commandes utilisateur à exécuter sur un terminal pour assembler l'image.

avions de contrôle hébergés

Fonctionnalité d'OpenShift Container Platform qui permet d'héberger un plan de contrôle sur le cluster OpenShift Container Platform à partir de son plan de données et de ses travailleurs. Ce modèle effectue les actions suivantes :

- Optimiser les coûts d'infrastructure requis pour les plans de contrôle.
- Améliorer le temps de création des clusters.
- Permettre l'hébergement du plan de contrôle en utilisant les primitives de haut niveau natives de Kubernetes. Par exemple, les déploiements, les ensembles avec état.
- Permettre une forte segmentation du réseau entre le plan de contrôle et les charges de travail.

les déploiements de nuages hybrides

Des déploiements qui offrent une plateforme cohérente dans les environnements bare metal, virtuels, privés et publics. Cela permet de gagner en rapidité, en agilité et en portabilité.

Allumage

Utilitaire utilisé par RHCOS pour manipuler les disques lors de la configuration initiale. Il effectue des tâches courantes sur les disques, notamment le partitionnement des disques, le formatage des partitions, l'écriture de fichiers et la configuration des utilisateurs.

l'infrastructure fournie par l'installateur

Le programme d'installation déploie et configure l'infrastructure sur laquelle le cluster fonctionne.

kubelet

Un agent de nœud primaire qui s'exécute sur chaque nœud du cluster pour s'assurer que les conteneurs s'exécutent dans un pod.

manifeste kubernetes

Spécifications d'un objet API Kubernetes dans un format JSON ou YAML. Un fichier de configuration peut inclure des déploiements, des cartes de configuration, des secrets, des ensembles de démons.

Machine Config Daemon (MCD)

Un démon qui vérifie régulièrement que les nœuds n'ont pas changé de configuration.

Opérateur de configuration de machine (MCO)

Un opérateur qui applique la nouvelle configuration aux machines de votre cluster.

pools de configuration de machines (MCP)

Groupe de machines, telles que les composants du plan de contrôle ou les charges de travail des utilisateurs, qui sont basées sur les ressources qu'elles gèrent.

métadonnées

Informations supplémentaires sur les artefacts de déploiement de clusters.

microservices

Une approche de l'écriture de logiciels. Les applications peuvent être séparées en composants les plus petits, indépendants les uns des autres, en utilisant des microservices.

registre miroir

Un registre qui contient le miroir des images d'OpenShift Container Platform.

applications monolithiques

Applications autonomes, construites et conditionnées en une seule pièce.

espaces nominatifs

Un espace de noms isole des ressources système spécifiques qui sont visibles par tous les processus. À l'intérieur d'un espace de noms, seuls les processus membres de cet espace peuvent voir ces ressources.

la mise en réseau

Informations sur le réseau du cluster OpenShift Container Platform.

nœud

Une machine de travail dans le cluster OpenShift Container Platform. Un nœud est soit une machine virtuelle (VM), soit une machine physique.

Service de mise à jour de la plateforme OpenShift Container (OSUS)

Pour les clusters disposant d'un accès à Internet, Red Hat Enterprise Linux (RHEL) fournit des mises à jour over-the-air en utilisant un service de mise à jour OpenShift Container Platform en tant que service hébergé situé derrière des API publiques.

OpenShift CLI (oc)

Un outil de ligne de commande pour exécuter des commandes OpenShift Container Platform sur le terminal.

OpenShift Dédié

Une offre de plateforme de conteneurs RHEL OpenShift gérée sur Amazon Web Services (AWS) et Google Cloud Platform (GCP). OpenShift Dedicated se concentre sur la création et la mise à l'échelle d'applications.

Registre d'images OpenShift

Un registre fourni par OpenShift Container Platform pour gérer les images.

Opérateur

La méthode préférée pour empaqueter, déployer et gérer une application Kubernetes dans un cluster OpenShift Container Platform. Un opérateur prend les connaissances opérationnelles humaines et les encode dans un logiciel qui est emballé et partagé avec les clients.

OperatorHub

Une plateforme qui contient divers opérateurs OpenShift Container Platform à installer.

Gestionnaire du cycle de vie des opérateurs (OLM)

OLM vous aide à installer, mettre à jour et gérer le cycle de vie des applications natives Kubernetes. OLM est une boîte à outils open source conçue pour gérer les opérateurs de manière efficace, automatisée et évolutive.

OSTree

Un système de mise à jour pour les systèmes d'exploitation basés sur Linux qui effectue des mises à jour atomiques d'arborescences complètes de systèmes de fichiers. OSTree suit les modifications significatives apportées à l'arborescence du système de fichiers à l'aide d'un magasin d'objets adressables et est conçu pour compléter les systèmes de gestion de paquets existants.

les mises à jour OTA (over-the-air)

Le service de mise à jour d'OpenShift Container Platform (OSUS) fournit des mises à jour over-the-air à OpenShift Container Platform, y compris Red Hat Enterprise Linux CoreOS (RHCOS).

nacelle

Un ou plusieurs conteneurs avec des ressources partagées, telles que le volume et les adresses IP, fonctionnant dans votre cluster OpenShift Container Platform. Un pod est la plus petite unité de calcul définie, déployée et gérée.

registre privé

OpenShift Container Platform peut utiliser n'importe quel serveur mettant en œuvre l'API de registre d'images de conteneurs comme source de l'image, ce qui permet aux développeurs de pousser et de tirer leurs images de conteneurs privées.

registre public

OpenShift Container Platform peut utiliser n'importe quel serveur mettant en œuvre l'API de registre d'images de conteneurs comme source de l'image, ce qui permet aux développeurs de pousser et de tirer leurs images de conteneurs publics.

RHEL OpenShift Container Platform Cluster Manager

Un service géré où vous pouvez installer, modifier, exploiter et mettre à niveau vos clusters OpenShift Container Platform.

RHEL Quay Container Registry

Un registre de conteneurs Quay.io qui fournit la plupart des images de conteneurs et des opérateurs aux clusters OpenShift Container Platform.

contrôleurs de réplication

Une ressource qui indique combien de répliques de pods sont nécessaires pour fonctionner en même temps.

le contrôle d'accès basé sur les rôles (RBAC)

Contrôle de sécurité essentiel pour garantir que les utilisateurs et les charges de travail des clusters n'ont accès qu'aux ressources nécessaires à l'exécution de leur rôle.

itinéraire

Les routes exposent un service permettant l'accès réseau aux pods à partir d'utilisateurs et d'applications extérieurs à l'instance d'OpenShift Container Platform.

mise à l'échelle

L'augmentation ou la diminution de la capacité des ressources.

service

Un service expose une application en cours d'exécution sur un ensemble de pods.

Image source-image (S2I)

Une image créée sur la base du langage de programmation du code source de l'application dans OpenShift Container Platform pour déployer des applications.

stockage

OpenShift Container Platform prend en charge de nombreux types de stockage, à la fois pour les fournisseurs sur site et dans le nuage. Vous pouvez gérer le stockage des conteneurs pour les données persistantes et non persistantes dans un cluster OpenShift Container Platform.

Télémétrie

Un composant pour collecter des informations telles que la taille, la santé et l'état de OpenShift Container Platform.

modèle

Un template décrit un ensemble d'objets qui peuvent être paramétrés et traités pour produire une liste d'objets à créer par OpenShift Container Platform.

l'infrastructure fournie par l'utilisateur

Vous pouvez installer OpenShift Container Platform sur l'infrastructure que vous fournissez. Vous pouvez utiliser le programme d'installation pour générer les ressources nécessaires au provisionnement de l'infrastructure du cluster, créer l'infrastructure du cluster, puis déployer le cluster sur l'infrastructure que vous avez fournie.

console web

Une interface utilisateur (UI) pour gérer OpenShift Container Platform.

nœud de travail

Nœuds chargés d'exécuter les charges de travail pour les utilisateurs de la grappe. Les nœuds de travail sont également appelés nœuds de calcul.

Ressources complémentaires

- Pour plus d'informations sur la mise en réseau, voir OpenShift Container Platform networking.
- Pour plus d'informations sur le stockage, voir OpenShift Container Platform storage.
- Pour plus d'informations sur l'authentification, voir OpenShift Container Platform authentication.
- Pour plus d'informations sur Operator Lifecycle Manager (OLM), voir OLM.
- Pour plus d'informations sur la journalisation, voir OpenShift Container Platform Logging.
- Pour plus d'informations sur les mises à jour OTA, voir Mise à jour des clusters OpenShift Container Platform.

1.2. A PROPOS DE L'INSTALLATION ET DES MISES À JOUR

En tant qu'administrateur de cluster, vous pouvez utiliser le programme d'installation d' OpenShift Container Platform pour installer et déployer un cluster en utilisant l'une des méthodes suivantes :

- Infrastructure fournie par l'installateur
- Infrastructure fournie par l'utilisateur

1.3. À PROPOS DU PLAN DE CONTRÔLE

Le plan de contrôle gère les nœuds de travail et les pods de votre cluster. Vous pouvez configurer les nœuds à l'aide de pools de configuration de machines (MCP). Les MCP sont des groupes de machines, tels que les composants du plan de contrôle ou les charges de travail des utilisateurs, qui sont basés sur

les ressources qu'ils gèrent. OpenShift Container Platform attribue différents rôles aux hôtes. Ces rôles définissent la fonction d'une machine dans un cluster. Le cluster contient des définitions pour les types de rôles standard de plan de contrôle et de travailleur.

Vous pouvez utiliser les opérateurs pour emballer, déployer et gérer les services sur le plan de contrôle. Les opérateurs sont des composants importants dans OpenShift Container Platform car ils fournissent les services suivants :

- Effectuer des contrôles de santé
- Fournir des moyens de surveiller les applications
- Gérer les mises à jour en direct
- S'assurer que les applications restent dans l'état spécifié

1.4. A PROPOS DES APPLICATIONS CONTENEURISÉES POUR LES DÉVELOPPEURS

En tant que développeur, vous pouvez utiliser différents outils, méthodes et formats pour développer votre application conteneurisée en fonction de vos besoins spécifiques, par exemple :

- Utiliser divers outils de construction, images de base et options de registre pour créer une application conteneurisée simple.
- Utilisez les composants de soutien tels que OperatorHub et les modèles pour développer votre application.
- Empaqueter et déployer votre application en tant qu'opérateur.

Vous pouvez également créer un manifeste Kubernetes et le stocker dans un dépôt Git. Kubernetes fonctionne avec des unités de base appelées pods. Un pod est une instance unique d'un processus en cours d'exécution dans votre cluster. Les pods peuvent contenir un ou plusieurs conteneurs. Vous pouvez créer un service en regroupant un ensemble de pods et leurs politiques d'accès. Les services fournissent des adresses IP internes permanentes et des noms d'hôtes que d'autres applications peuvent utiliser lors de la création et de la destruction des pods. Kubernetes définit les charges de travail en fonction du type de votre application.

1.5. À PROPOS DE RED HAT ENTERPRISE LINUX COREOS (RHCOS) ET IGNITION

En tant qu'administrateur de cluster, vous pouvez effectuer les tâches suivantes de Red Hat Enterprise Linux CoreOS (RHCOS) :

- Découvrez la nouvelle génération de systèmes d'exploitation de conteneurs à usage unique .
- Choisir comment configurer Red Hat Enterprise Linux CoreOS (RHCOS)
- Choisissez comment déployer Red Hat Enterprise Linux CoreOS (RHCOS) :
  - Déploiement fourni par l'installateur
  - Déploiement à la demande de l'utilisateur

Le programme d'installation d'OpenShift Container Platform crée les fichiers de configuration Ignition

dont vous avez besoin pour déployer votre cluster. Red Hat Enterprise Linux CoreOS (RHCOS) utilise Ignition lors de la configuration initiale pour effectuer des tâches de disque communes, telles que le partitionnement, le formatage, l'écriture de fichiers et la configuration des utilisateurs. Lors du premier démarrage, Ignition lit sa configuration à partir du média d'installation ou de l'emplacement que vous avez spécifié et applique la configuration aux machines.

Vous pouvez apprendre comment fonctionne Ignition, le processus pour une machine Red Hat Enterprise Linux CoreOS (RHCOS) dans un cluster OpenShift Container Platform, voir les fichiers de configuration d'Ignition, et modifier la configuration d'Ignition après une installation.

1.6. A PROPOS DES PLUGINS D'ADMISSION

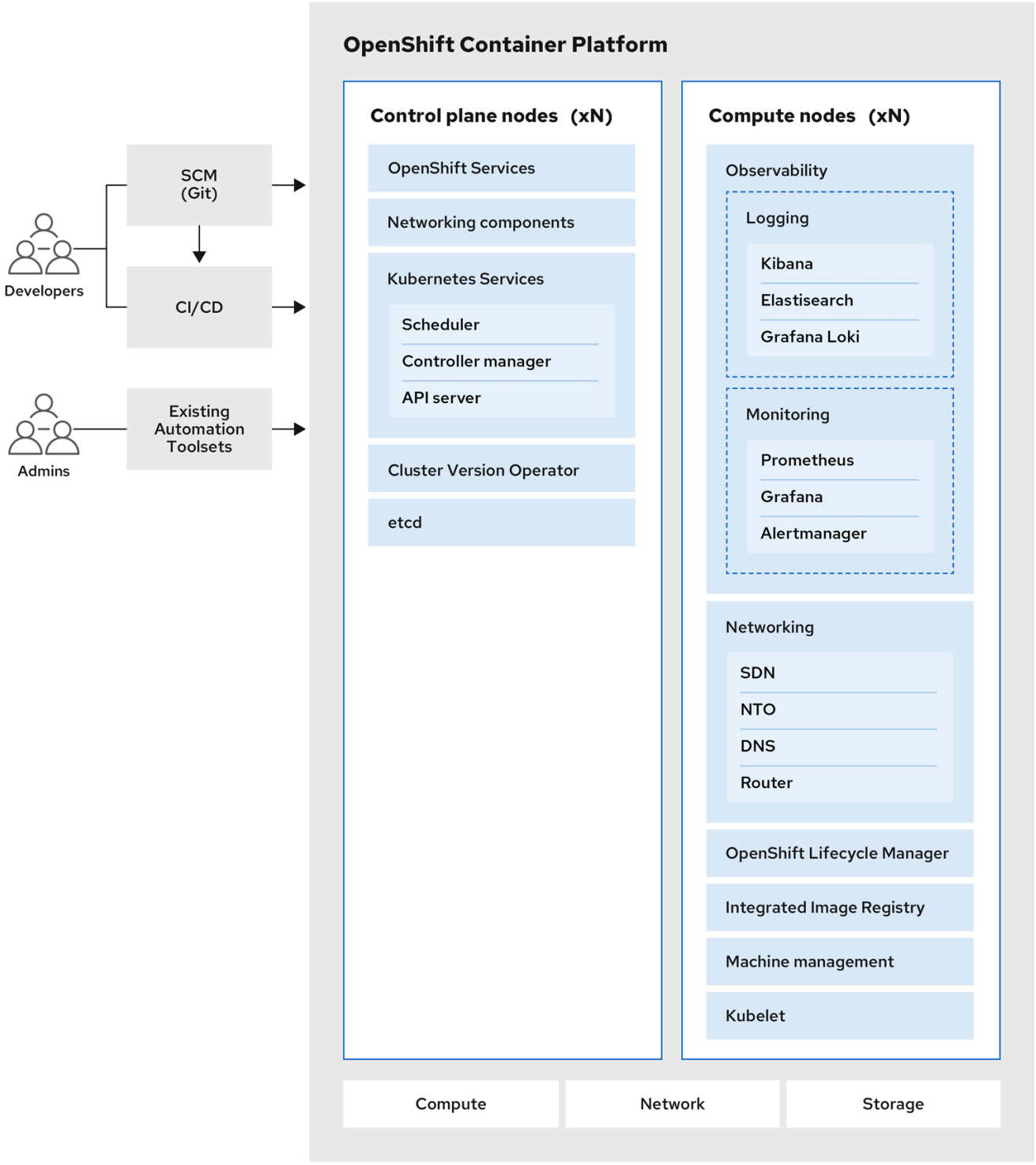
Vous pouvez utiliser les plugins d'admission pour réguler le fonctionnement d'OpenShift Container Platform. Une fois qu'une demande de ressource est authentifiée et autorisée, les plugins d'admission interceptent la demande de ressource vers l'API maître afin de valider les demandes de ressources et de s'assurer que les politiques de mise à l'échelle sont respectées. Les plugins d'admission sont utilisés pour appliquer les politiques de sécurité, les limitations de ressources ou les exigences de configuration.

CHAPITRE 2. ARCHITECTURE DE LA PLATEFORME DE CONTENEURS OPENSHIFT

2.1. INTRODUCTION À OPENSHIFT CONTAINER PLATFORM

OpenShift Container Platform est une plateforme de développement et d'exécution d'applications conteneurisées. Elle est conçue pour permettre aux applications et aux centres de données qui les prennent en charge de passer de quelques machines et applications à des milliers de machines desservant des millions de clients.

Avec son fondement dans Kubernetes, OpenShift Container Platform incorpore la même technologie qui sert de moteur aux télécommunications massives, à la vidéo en continu, aux jeux, à la banque et à d'autres applications. Sa mise en œuvre dans les technologies ouvertes Red Hat vous permet d'étendre vos applications conteneurisées au-delà d'un seul cloud à des environnements sur site et multicloud.



2.1.1. À propos de Kubernetes

Bien que les images de conteneurs et les conteneurs qui s'exécutent à partir d'elles soient les principaux éléments constitutifs du développement d'applications modernes, leur exécution à grande échelle nécessite un système de distribution fiable et flexible. Kubernetes est la norme de facto pour l'orchestration des conteneurs.

Kubernetes est un moteur d'orchestration de conteneurs open source permettant d'automatiser le déploiement, la mise à l'échelle et la gestion des applications conteneurisées. Le concept général de Kubernetes est assez simple :

- Commencez par un ou plusieurs nœuds de travail pour exécuter les charges de travail des conteneurs.
- Gérer le déploiement de ces charges de travail à partir d'un ou plusieurs nœuds du plan de contrôle.
- Envelopper les conteneurs dans une unité de déploiement appelée pod. L'utilisation de pods fournit des métadonnées supplémentaires avec le conteneur et offre la possibilité de regrouper plusieurs conteneurs dans une seule entité de déploiement.
- Créer des types d'actifs particuliers. Par exemple, les services sont représentés par un ensemble de pods et une politique qui définit comment y accéder. Cette politique permet aux conteneurs de se connecter aux services dont ils ont besoin, même s'ils ne disposent pas des adresses IP spécifiques de ces services. Les contrôleurs de réplication sont une autre ressource spéciale qui indique combien de répliques de pods sont nécessaires pour fonctionner à un moment donné. Vous pouvez utiliser cette capacité pour adapter automatiquement votre application à la demande actuelle.

En quelques années seulement, Kubernetes a connu une adoption massive dans le cloud et sur site. Le modèle de développement open source permet à de nombreuses personnes d'étendre Kubernetes en mettant en œuvre différentes technologies pour des composants tels que le réseau, le stockage et l'authentification.

2.1.2. Les avantages des applications conteneurisées

L'utilisation d'applications conteneurisées présente de nombreux avantages par rapport aux méthodes de déploiement traditionnelles. Alors que les applications étaient autrefois censées être installées sur des systèmes d'exploitation comprenant toutes leurs dépendances, les conteneurs permettent à une application de transporter ses dépendances avec elle. La création d'applications conteneurisées offre de nombreux avantages.

2.1.2.1. Avantages du système d'exploitation

Les conteneurs utilisent de petits systèmes d'exploitation Linux dédiés sans noyau. Leur système de fichiers, leur réseau, leurs cgroups, leurs tables de processus et leurs espaces de noms sont distincts du système Linux hôte, mais les conteneurs peuvent s'intégrer aux hôtes de manière transparente si nécessaire. Le fait d'être basé sur Linux permet aux conteneurs d'utiliser tous les avantages qui viennent avec le modèle de développement open source d'innovation rapide.

Chaque conteneur utilisant un système d'exploitation dédié, vous pouvez déployer sur le même hôte des applications nécessitant des dépendances logicielles conflictuelles. Chaque conteneur transporte ses propres logiciels dépendants et gère ses propres interfaces, telles que les réseaux et les systèmes de fichiers, de sorte que les applications n'ont jamais besoin d'entrer en concurrence pour ces actifs.

2.1.2.2. Avantages en termes de déploiement et de mise à l'échelle

Si vous utilisez des mises à jour en continu entre les versions majeures de votre application, vous pouvez améliorer continuellement vos applications sans temps d'arrêt tout en conservant la compatibilité avec la version actuelle.

Vous pouvez également déployer et tester une nouvelle version d'une application parallèlement à la version existante. Si le conteneur passe les tests avec succès, il suffit de déployer d'autres nouveaux conteneurs et de supprimer les anciens.

Étant donné que toutes les dépendances logicielles d'une application sont résolues dans le conteneur lui-même, vous pouvez utiliser un système d'exploitation standardisé sur chaque hôte de votre centre de

données. Il n'est pas nécessaire de configurer un système d'exploitation spécifique pour chaque hôte d'application. Lorsque votre centre de données a besoin de plus de capacité, vous pouvez déployer un autre système hôte générique.

De même, la mise à l'échelle des applications conteneurisées est simple. OpenShift Container Platform offre une méthode simple et standard de mise à l'échelle de tout service conteneurisé. Par exemple, si vous créez des applications sous la forme d'un ensemble de microservices plutôt que de grandes applications monolithiques, vous pouvez mettre à l'échelle les microservices individuellement pour répondre à la demande. Cette capacité vous permet de ne faire évoluer que les services nécessaires au lieu de l'ensemble de l'application, ce qui peut vous permettre de répondre aux demandes de l'application tout en utilisant un minimum de ressources.

2.1.3. Aperçu de la plateforme OpenShift Container

OpenShift Container Platform apporte à Kubernetes des améliorations prêtes pour l'entreprise, notamment les suivantes :

- Déploiements dans le nuage hybride. Vous pouvez déployer des clusters OpenShift Container Platform sur diverses plateformes de cloud public ou dans votre centre de données.
- Technologie Red Hat intégrée. Les principaux composants d'OpenShift Container Platform proviennent de Red Hat Enterprise Linux (RHEL) et des technologies Red Hat connexes. OpenShift Container Platform bénéficie des initiatives intenses de test et de certification des logiciels de qualité d'entreprise de Red Hat.
- Modèle de développement à source ouverte. Le développement est réalisé de manière ouverte et le code source est disponible dans des dépôts de logiciels publics. Cette collaboration ouverte favorise l'innovation et le développement rapides.

Bien que Kubernetes excelle dans la gestion de vos applications, il ne spécifie ni ne gère les exigences au niveau de la plateforme ou les processus de déploiement. Des outils et des processus de gestion de plateforme puissants et flexibles sont des avantages importants qu'offre OpenShift Container Platform 4.12. Les sections suivantes décrivent certaines caractéristiques et avantages uniques d'OpenShift Container Platform.

2.1.3.1. Système d'exploitation personnalisé

OpenShift Container Platform utilise Red Hat Enterprise Linux CoreOS (RHCOS), un système d'exploitation orienté conteneur spécialement conçu pour l'exécution d'applications conteneurisées à partir d'OpenShift Container Platform et fonctionne avec de nouveaux outils pour offrir une installation rapide, une gestion basée sur l'opérateur et des mises à niveau simplifiées.

Le RHCOS comprend

- Ignition, qu'OpenShift Container Platform utilise comme configuration système de premier démarrage pour démarrer et configurer les machines.
- CRI-O, une implémentation native du runtime de conteneur Kubernetes qui s'intègre étroitement avec le système d'exploitation pour fournir une expérience Kubernetes efficace et optimisée. CRI-O permet d'exécuter, d'arrêter et de redémarrer les conteneurs. Il remplace entièrement le moteur de conteneurs Docker, qui était utilisé dans OpenShift Container Platform 3.
- Kubelet, le principal agent de nœud pour Kubernetes qui est responsable du lancement et de la surveillance des conteneurs.

Dans OpenShift Container Platform 4.12, vous devez utiliser RHCOS pour toutes les machines du plan de contrôle, mais vous pouvez utiliser Red Hat Enterprise Linux (RHEL) comme système d'exploitation pour les machines de calcul, qui sont également connues sous le nom de machines de travail. Si vous choisissez d'utiliser les machines de calcul RHEL, vous devrez effectuer davantage de maintenance système que si vous utilisez RHCOS pour toutes les machines du cluster.

2.1.3.2. Processus d'installation et de mise à jour simplifié

Avec OpenShift Container Platform 4.12, si vous disposez d'un compte avec les bonnes autorisations, vous pouvez déployer un cluster de production dans les clouds pris en charge en exécutant une seule commande et en fournissant quelques valeurs. Vous pouvez également personnaliser votre installation dans le nuage ou installer votre cluster dans votre centre de données si vous utilisez une plateforme prise en charge.

Pour les clusters qui utilisent RHCOS pour toutes les machines, la mise à jour ou la mise à niveau d'OpenShift Container Platform est un processus simple et hautement automatisé. Comme OpenShift Container Platform contrôle entièrement les systèmes et les services qui s'exécutent sur chaque machine, y compris le système d'exploitation lui-même, à partir d'un plan de contrôle central, les mises à niveau sont conçues pour devenir des événements automatiques. Si votre cluster contient des machines de travail RHEL, le plan de contrôle bénéficie du processus de mise à jour rationalisé, mais vous devez effectuer davantage de tâches pour mettre à niveau les machines RHEL.

2.1.3.3. Autres caractéristiques clés

Les opérateurs sont à la fois l'unité fondamentale de la base de code d'OpenShift Container Platform 4.12 et un moyen pratique de déployer des applications et des composants logiciels pour que vos applications les utilisent. Dans OpenShift Container Platform, les opérateurs servent de fondation à la plateforme et éliminent le besoin de mises à niveau manuelles des systèmes d'exploitation et des applications du plan de contrôle. Les opérateurs d'OpenShift Container Platform tels que le Cluster Version Operator et le Machine Config Operator permettent une gestion simplifiée, à l'échelle du cluster, de ces composants critiques.

L'Operator Lifecycle Manager (OLM) et l'OperatorHub permettent de stocker et de distribuer des opérateurs aux personnes qui développent et déploient des applications.

Le Red Hat Quay Container Registry est un registre de conteneurs Quay.io qui fournit la plupart des images de conteneurs et des opérateurs aux clusters OpenShift Container Platform. Quay.io est une version du registre public de Red Hat Quay qui stocke des millions d'images et de tags.

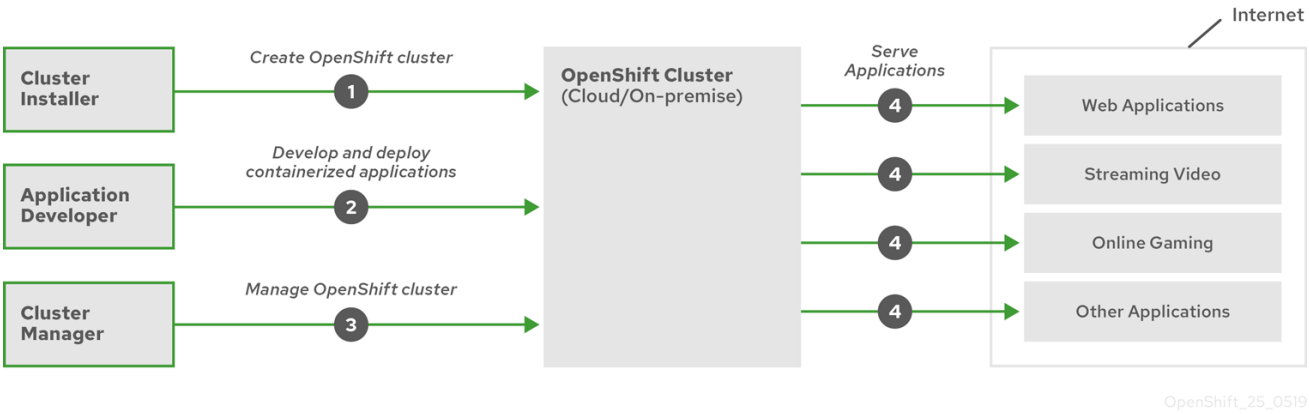
Parmi les autres améliorations apportées à Kubernetes dans OpenShift Container Platform, citons les améliorations apportées au réseau défini par logiciel (SDN), à l'authentification, à l'agrégation des journaux, à la surveillance et au routage. OpenShift Container Platform offre également une console web complète et l'interface CLI (**oc**) personnalisée d'OpenShift.

2.1.3.4. Cycle de vie d'OpenShift Container Platform

La figure suivante illustre le cycle de vie de base d'OpenShift Container Platform :

- Créer un cluster OpenShift Container Platform
- Gestion de la grappe
- Développement et déploiement d'applications
- Extension des applications

Figure 2.1. Vue d'ensemble de la plateforme OpenShift Container Platform



2.1.4. Accès à l'internet pour OpenShift Container Platform

Dans OpenShift Container Platform 4.12, vous devez avoir accès à Internet pour installer votre cluster.

Vous devez disposer d'un accès à l'internet pour :

- Accédez à OpenShift Cluster Manager Hybrid Cloud Console pour télécharger le programme d'installation et effectuer la gestion des abonnements. Si le cluster dispose d'un accès internet et que vous ne désactivez pas Telemetry, ce service donne automatiquement des droits à votre cluster.
- Accédez à Quay.io pour obtenir les paquets nécessaires à l'installation de votre cluster.
- Obtenir les paquets nécessaires pour effectuer les mises à jour de la grappe.

IMPORTANT

Si votre cluster ne peut pas avoir d'accès direct à l'internet, vous pouvez effectuer une installation en réseau restreint sur certains types d'infrastructure que vous fournissez. Au cours de ce processus, vous téléchargez le contenu requis et l'utilisez pour remplir un registre miroir avec les paquets d'installation. Avec certains types d'installation, l'environnement dans lequel vous installez votre cluster ne nécessite pas d'accès à Internet. Avant de mettre à jour le cluster, vous mettez à jour le contenu du registre miroir.

CHAPITRE 3. INSTALLATION ET MISE À JOUR

3.1. A PROPOS DE L'INSTALLATION D'OPENSHIFT CONTAINER PLATFORM

Le programme d'installation d'OpenShift Container Platform propose quatre méthodes pour déployer un cluster :

- **Interactive**: Vous pouvez déployer un cluster à l'aide du programme d'installation assistée basé sur le web. C'est l'approche recommandée pour les clusters avec des réseaux connectés à l'internet. L'installateur assisté est le moyen le plus simple d'installer OpenShift Container Platform, il fournit des valeurs par défaut intelligentes et effectue des validations avant l'installation du cluster. Il fournit également une API RESTful pour l'automatisation et les scénarios de configuration avancés.

- **Local Agent-based**: Vous pouvez déployer un cluster localement avec le programme d'installation basé sur un agent pour les réseaux fermés ou restreints. Il offre de nombreux avantages par rapport à l'installateur assisté, mais vous devez d'abord télécharger et configurer l'installateur basé sur un agent. La configuration s'effectue à l'aide d'une interface de ligne de commande. Cette approche est idéale pour les réseaux fermés ou restreints.

- **Automated**: Vous pouvez déployer une grappe sur une infrastructure fournie par le programme d'installation et la grappe qu'elle maintient. Le programme d'installation utilise le contrôleur de gestion de carte de base (BMC) de chaque hôte de cluster pour le provisionnement. Vous pouvez déployer des clusters avec des réseaux connectés ou restreints.

- **Full control**: Vous pouvez déployer une grappe sur une infrastructure que vous préparez et entretenez, ce qui offre un maximum de possibilités de personnalisation. Vous pouvez déployer des clusters avec des réseaux connectés ou restreints.

Les grappes présentent les caractéristiques suivantes :

- Une infrastructure hautement disponible, sans point de défaillance unique, est disponible par défaut.

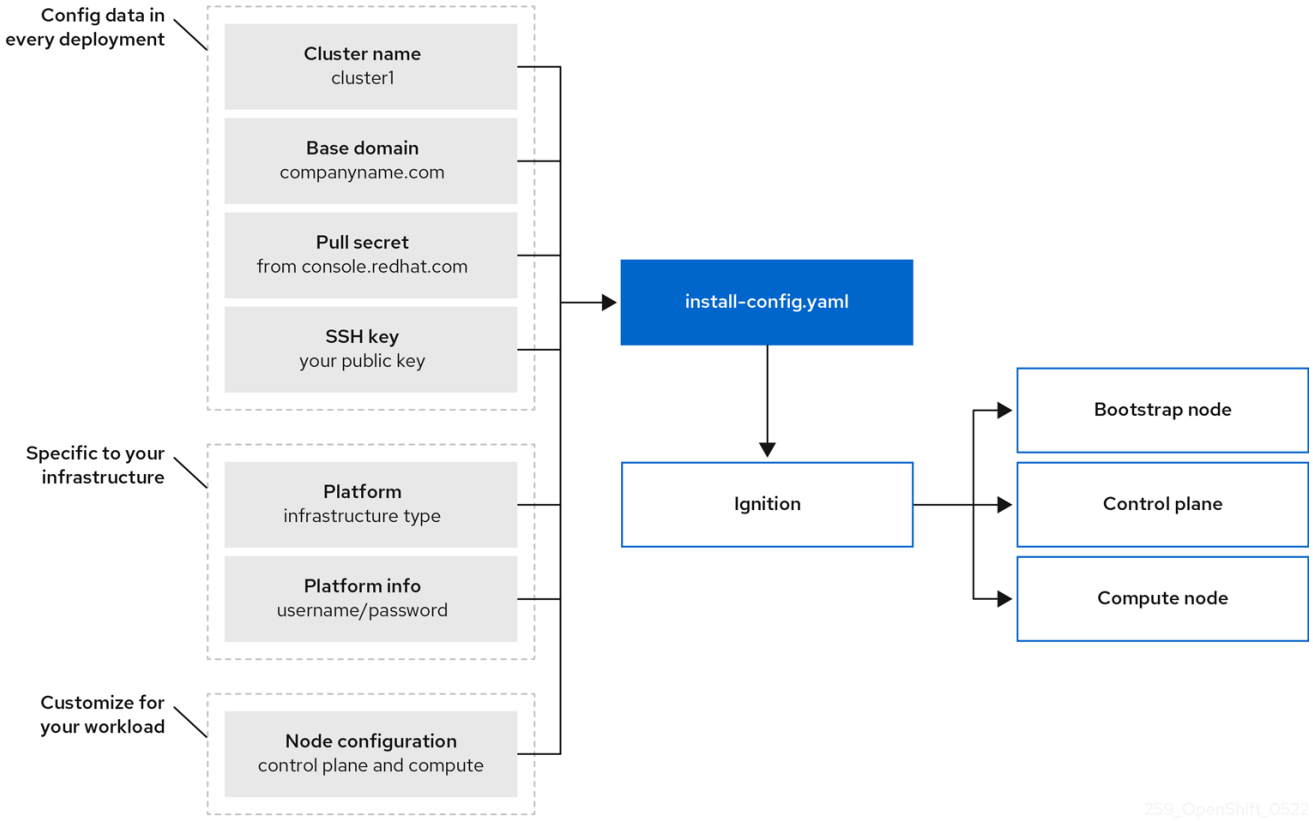
- Les administrateurs gardent le contrôle sur les mises à jour qui sont appliquées et sur le moment où elles le sont.

3.1.1. A propos du programme d'installation

Vous pouvez utiliser le programme d'installation pour déployer chaque type de cluster. Le programme d'installation génère des ressources principales telles que les fichiers de configuration Ignition pour le bootstrap, le plan de contrôle (maître) et les machines de travail. Vous pouvez démarrer un cluster OpenShift Container Platform avec ces trois configurations et une infrastructure correctement configurée.

Le programme d'installation d'OpenShift Container Platform utilise un ensemble de cibles et de dépendances pour gérer les installations de clusters. Le programme d'installation a un ensemble de cibles qu'il doit atteindre, et chaque cible a un ensemble de dépendances. Comme chaque objectif ne concerne que ses propres dépendances, le programme d'installation peut agir pour atteindre plusieurs objectifs en parallèle, l'objectif final étant un cluster en cours d'exécution. Le programme d'installation reconnaît et utilise les composants existants au lieu d'exécuter des commandes pour les créer à nouveau, car le programme respecte les dépendances.

Figure 3.1. Cibles et dépendances pour l'installation d'OpenShift Container Platform



3.1.2. À propos de Red Hat Enterprise Linux CoreOS (RHCOS)

Après l'installation, chaque machine du cluster utilise Red Hat Enterprise Linux CoreOS (RHCOS) comme système d'exploitation. RHCOS est la version d'hôte de conteneur immuable de Red Hat Enterprise Linux (RHEL) et comporte un noyau RHEL avec SELinux activé par défaut. Il inclut **kubelet**, qui est l'agent de nœud Kubernetes, et le runtime de conteneur CRI-O, qui est optimisé pour Kubernetes.

Chaque machine du plan de contrôle dans un cluster OpenShift Container Platform 4.12 doit utiliser RHCOS, qui comprend un outil critique de provisionnement au premier démarrage appelé Ignition. Cet outil permet au cluster de configurer les machines. Les mises à jour du système d'exploitation sont fournies sous la forme d'une image de conteneur amorçable, utilisant **OSTree** comme backend, qui est déployée dans le cluster par le Machine Config Operator. Les modifications réelles du système d'exploitation sont effectuées sur place sur chaque machine en tant qu'opération atomique à l'aide de **rpm-ostree**. Ensemble, ces technologies permettent à OpenShift Container Platform de gérer le système d'exploitation comme n'importe quelle autre application sur le cluster, en effectuant des mises à jour sur place qui maintiennent l'ensemble de la plateforme à jour. Ces mises à jour sur place peuvent réduire la charge des équipes d'exploitation.

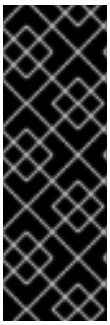
Si vous utilisez RHCOS comme système d'exploitation pour toutes les machines de la grappe, celle-ci gère tous les aspects de ses composants et de ses machines, y compris le système d'exploitation. Pour cette raison, seuls le programme d'installation et l'opérateur de configuration des machines peuvent modifier les machines. Le programme d'installation utilise les fichiers de configuration Ignition pour définir l'état exact de chaque machine, et le Machine Config Operator effectue d'autres modifications sur les machines, telles que l'application de nouveaux certificats ou de nouvelles clés, après l'installation.

3.1.3. Plateformes prises en charge pour les clusters OpenShift Container Platform

Dans OpenShift Container Platform 4.12, vous pouvez installer un cluster qui utilise une infrastructure fournie par l'installateur sur les plateformes suivantes :

- Amazon Web Services (AWS)
- Google Cloud Platform (GCP)
- Microsoft Azure
- Microsoft Azure Stack Hub
- Red Hat OpenStack Platform (RHOSP) versions 16.1 et 16.2
  - La dernière version d'OpenShift Container Platform supporte à la fois la dernière version RHOSP longue durée et la version intermédiaire. Pour une compatibilité complète avec les versions RHOSP, voir la matrice de prise en charge d'OpenShift Container Platform sur RHOSP.
- IBM Cloud VPC
- Nutanix
- Red Hat Virtualization (RHV)
- VMware vSphere
- Cloud VMware (VMC) sur AWS
- Nuage d'Alibaba
- Métal nu

Pour ces clusters, toutes les machines, y compris l'ordinateur sur lequel vous exécutez le processus d'installation, doivent avoir un accès direct à Internet afin d'extraire des images pour les conteneurs de la plate-forme et de fournir des données télémétriques à Red Hat.



IMPORTANT

Après l'installation, les modifications suivantes ne sont pas prises en charge :

- Mélanger les plates-formes des fournisseurs d'informatique en nuage
- Mélange de composants de fournisseurs de cloud, comme l'utilisation d'un cadre de stockage persistant d'une plate-forme différente de celle sur laquelle le cluster est installé

Dans OpenShift Container Platform 4.12, vous pouvez installer un cluster qui utilise l'infrastructure fournie par l'utilisateur sur les plateformes suivantes :

- AWS
- L'azur
- Hub Azure Stack
- PCG

- RHOSP versions 16.1 et 16.2
- RHV
- VMware vSphere
- Cloud VMware sur AWS
- Métal nu
- IBM zSystems ou IBM® LinuxONE
- IBM Power

Selon les cas pris en charge pour la plateforme, les installations sur une infrastructure fournie par l'utilisateur vous permettent d'exécuter des machines avec un accès complet à Internet, de placer votre cluster derrière un proxy ou d'effectuer une installation sur *restricted network installation*. Dans une installation en réseau restreint, vous pouvez télécharger les images nécessaires à l'installation d'un cluster, les placer dans un registre miroir et utiliser ces données pour installer votre cluster. Alors que vous avez besoin d'un accès à Internet pour extraire les images des conteneurs de plateforme, avec une installation en réseau restreint sur vSphere ou une infrastructure bare metal, vos machines de cluster n'ont pas besoin d'un accès direct à Internet.

La page OpenShift Container Platform 4.x Tested Integrations contient des détails sur les tests d'intégration pour différentes plateformes.

3.1.4. Processus d'installation

À l'exception de l'installateur assisté, lorsque vous installez un cluster OpenShift Container Platform, vous téléchargez le programme d'installation à partir de la page du fournisseur d'infrastructure approprié sur le site OpenShift Cluster Manager. Ce site gère :

- API REST pour les comptes
- Les jetons de registre, qui sont les secrets d'extraction que vous utilisez pour obtenir les composants requis
- L'enregistrement du cluster, qui associe l'identité du cluster à votre compte Red Hat afin de faciliter la collecte de données d'utilisation

Dans OpenShift Container Platform 4.12, le programme d'installation est un fichier binaire Go qui effectue une série de transformations de fichiers sur un ensemble de ressources. La façon dont vous interagissez avec le programme d'installation diffère en fonction de votre type d'installation.

- Pour déployer une grappe avec le programme d'installation assistée, vous devez configurer les paramètres de la grappe à l'aide du programme d'installation assistée. Il n'y a pas de programme d'installation à télécharger et à configurer. Une fois la configuration terminée, vous téléchargez une ISO de découverte et démarrez les machines du cluster avec cette image. Vous pouvez installer des clusters avec l'installateur assisté sur Nutanix, vSphere, et bare metal avec une intégration complète, et d'autres plateformes sans intégration. Si vous installez sur du bare metal, vous devez fournir toute l'infrastructure et les ressources du cluster, y compris le réseau, l'équilibrage de charge, le stockage et les machines individuelles du cluster.
- Pour déployer des grappes avec le programme d'installation basé sur un agent, vous devez d'abord télécharger le programme d'installation basé sur un agent. Ensuite, vous configurez la grappe et générez une image de découverte. Vous démarrez les machines de la grappe avec l'image de découverte, qui installe un agent qui communique avec le programme d'installation et

gère le provisionnement pour vous au lieu d'interagir avec le programme d'installation ou de configurer vous-même une machine de provisionnement. Vous devez fournir l'ensemble de l'infrastructure et des ressources du cluster, y compris le réseau, l'équilibrage de la charge, le stockage et les machines individuelles du cluster. Cette approche est idéale pour les environnements réseau restreints ou fermés.

- Pour les clusters dont l'infrastructure est fournie par l'installateur, vous déléguez le démarrage et le provisionnement de l'infrastructure au programme d'installation au lieu de le faire vous-même. Le programme d'installation crée tous les réseaux, machines et systèmes d'exploitation nécessaires à la prise en charge du cluster, sauf si vous procédez à une installation sur métal nu. Dans ce cas, vous devez fournir l'ensemble de l'infrastructure et des ressources du cluster, y compris la machine de démarrage, la mise en réseau, l'équilibrage de la charge, le stockage et les machines individuelles du cluster.

- Si vous fournissez et gérez l'infrastructure de votre cluster, vous devez fournir l'ensemble de l'infrastructure et des ressources du cluster, y compris la machine de démarrage, le réseau, l'équilibrage de la charge, le stockage et les machines individuelles du cluster.

Le programme d'installation utilise trois ensembles de fichiers lors de l'installation : un fichier de configuration d'installation nommé **install-config.yaml**, des manifestes Kubernetes et des fichiers de configuration Ignition pour vos types de machines.

IMPORTANT

Il est possible de modifier Kubernetes et les fichiers de configuration Ignition qui contrôlent le système d'exploitation RHCOS sous-jacent pendant l'installation. Cependant, aucune validation n'est disponible pour confirmer l'adéquation des modifications que vous apportez à ces objets. Si vous modifiez ces objets, vous risquez de rendre votre cluster non fonctionnel. En raison de ce risque, la modification des fichiers de configuration de Kubernetes et d'Ignition n'est pas prise en charge, à moins que vous ne suiviez des procédures documentées ou que l'assistance de Red Hat vous ait demandé de le faire.

Le fichier de configuration de l'installation est transformé en manifestes Kubernetes, puis les manifestes sont enveloppés dans des fichiers de configuration Ignition. Le programme d'installation utilise ces fichiers de configuration Ignition pour créer le cluster.

Les fichiers de configuration de l'installation sont tous élagués lorsque vous exécutez le programme d'installation. Veillez donc à sauvegarder tous les fichiers de configuration que vous souhaitez réutiliser.

IMPORTANT

Vous ne pouvez pas modifier les paramètres définis lors de l'installation, mais vous pouvez modifier de nombreux attributs du cluster après l'installation.

Le processus d'installation avec le programme d'installation assistée
L'installation à l'aide du programme d'installation assistée implique la création d'une configuration de cluster de manière interactive à l'aide de l'interface utilisateur Web ou de l'API RESTful. L'interface utilisateur du programme d'installation assistée vous invite à saisir les valeurs requises et fournit des valeurs par défaut raisonnables pour les autres paramètres, sauf si vous les modifiez dans l'interface utilisateur ou à l'aide de l'API. Le programme d'installation assistée génère une image de découverte que vous téléchargez et utilisez pour démarrer les machines du cluster. L'image installe RHCOS et un agent, et l'agent gère le provisionnement pour vous. Vous pouvez installer OpenShift Container Platform avec l'installateur assisté et l'intégration complète sur Nutanix, vSphere et bare metal, et sur d'autres plateformes sans intégration.

OpenShift Container Platform gère tous les aspects du cluster, y compris le système d'exploitation lui-même. Chaque machine démarre avec une configuration qui référence les ressources hébergées dans le cluster qu'elle rejoint. Cette configuration permet au cluster de se gérer lui-même au fur et à mesure que les mises à jour sont appliquées.

Si possible, utilisez cette fonction pour éviter d'avoir à télécharger et à configurer le programme d'installation basé sur un agent.

Le processus d'installation d'une infrastructure basée sur des agents
L'installation à l'aide d'un agent est similaire à l'utilisation du programme d'installation assistée, à ceci près que vous devez d'abord télécharger et installer le programme d'installation à l'aide d'un agent . L'installation à l'aide d'un agent est recommandée lorsque vous souhaitez bénéficier de tous les avantages de l'installateur assisté, mais que vous devez effectuer l'installation sur un réseau isolé ou déconnecté.

Si possible, utilisez cette fonctionnalité pour éviter d'avoir à créer une machine de provisionnement avec une VM de démarrage et de provisionner et maintenir l'infrastructure du cluster.

Le processus d'installation avec l'infrastructure fournie par l'installateur
Le type d'installation par défaut utilise une infrastructure fournie par l'installateur. Par défaut, le programme d'installation agit comme un assistant d'installation, vous demandant les valeurs qu'il ne peut pas déterminer lui-même et fournissant des valeurs par défaut raisonnables pour les paramètres restants. Vous pouvez également personnaliser le processus d'installation pour prendre en charge des scénarios d'infrastructure avancés. Le programme d'installation fournit l'infrastructure sous-jacente pour le cluster.

Vous pouvez installer soit un cluster standard, soit un cluster personnalisé. Dans le cas d'un cluster standard, vous fournissez les informations minimales nécessaires à l'installation du cluster. Avec un cluster personnalisé, vous pouvez spécifier plus de détails sur la plateforme, tels que le nombre de machines que le plan de contrôle utilise, le type de machine virtuelle que le cluster déploie, ou la plage CIDR pour le réseau de service Kubernetes.

Si possible, utilisez cette fonction pour éviter d'avoir à approvisionner et à maintenir l'infrastructure du cluster. Dans tous les autres environnements, vous utilisez le programme d'installation pour générer les ressources nécessaires à l'approvisionnement de votre infrastructure de cluster.

Avec les clusters d'infrastructure provisionnés par l'installateur, OpenShift Container Platform gère tous les aspects du cluster, y compris le système d'exploitation lui-même. Chaque machine démarre avec une configuration qui référence les ressources hébergées dans le cluster qu'elle rejoint. Cette configuration permet au cluster de se gérer lui-même au fur et à mesure que les mises à jour sont appliquées.

Le processus d'installation avec l'infrastructure fournie par l'utilisateur
Vous pouvez également installer OpenShift Container Platform sur une infrastructure que vous fournissez. Vous utilisez le programme d'installation pour générer les ressources nécessaires au provisionnement de l'infrastructure du cluster, créer l'infrastructure du cluster, puis déployer le cluster sur l'infrastructure que vous avez fournie.

Si vous n'utilisez pas l'infrastructure fournie par le programme d'installation, vous devez gérer et maintenir les ressources du cluster vous-même, y compris :

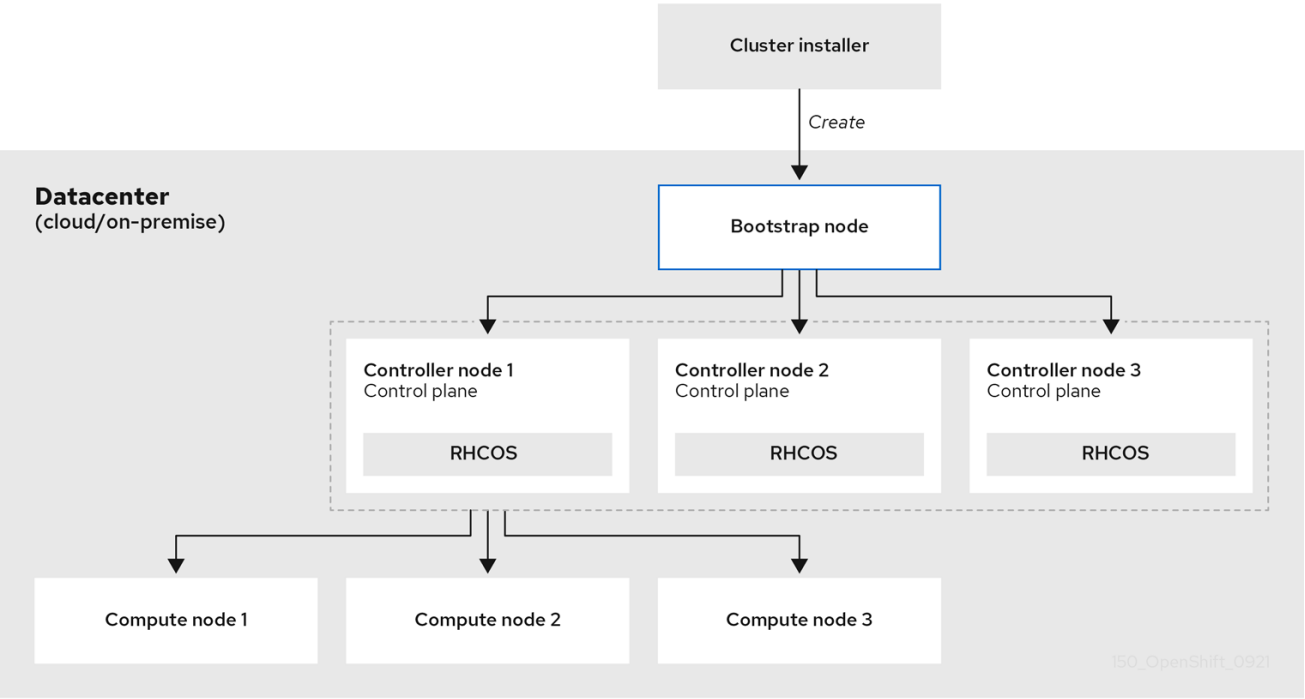
- L'infrastructure sous-jacente pour le plan de contrôle et les machines de calcul qui composent le cluster
- Équilibreurs de charge
- Mise en réseau du cluster, y compris les enregistrements DNS et les sous-réseaux requis

- Stockage pour l'infrastructure et les applications de la grappe

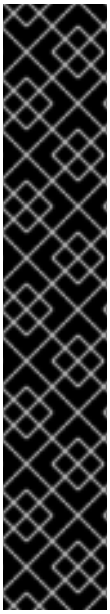
Si votre cluster utilise une infrastructure fournie par l'utilisateur, vous avez la possibilité d'ajouter des machines de calcul RHEL à votre cluster.

Détails de la procédure d'installation
Parce que chaque machine du cluster nécessite des informations sur le cluster lorsqu'elle est provisionnée, OpenShift Container Platform utilise une machine temporaire *bootstrap* lors de la configuration initiale pour fournir les informations nécessaires au plan de contrôle permanent. Elle démarre en utilisant un fichier de configuration Ignition qui décrit comment créer le cluster. La machine d'amorçage crée les machines du plan de contrôle qui constituent le plan de contrôle. Les machines du plan de contrôle créent ensuite les machines de calcul, également appelées machines de travail. La figure suivante illustre ce processus :

Figure 3.2. Création du bootstrap, du plan de contrôle et des machines de calcul



Après l'initialisation des machines de la grappe, la machine bootstrap est détruite. Tous les clusters utilisent le processus d'amorçage pour initialiser le cluster, mais si vous fournissez l'infrastructure pour votre cluster, vous devez effectuer de nombreuses étapes manuellement.



IMPORTANT

- Les fichiers de configuration d'Ignition générés par le programme d'installation contiennent des certificats qui expirent après 24 heures et qui sont renouvelés à ce moment-là. Si le cluster est arrêté avant le renouvellement des certificats et qu'il est redémarré après l'expiration des 24 heures, le cluster récupère automatiquement les certificats expirés. L'exception est que vous devez approuver manuellement les demandes de signature de certificat (CSR) de **node-bootstrapper** en attente pour récupérer les certificats de kubelet. Pour plus d'informations, consultez la documentation relative à *Recovering from expired control plane certificates*.

- Il est recommandé d'utiliser les fichiers de configuration Ignition dans les 12 heures suivant leur génération, car le certificat de 24 heures tourne entre 16 et 22 heures après l'installation du cluster. En utilisant les fichiers de configuration Ignition dans les 12 heures, vous pouvez éviter l'échec de l'installation si la mise à jour du certificat s'exécute pendant l'installation.

L'amorçage d'une grappe comprend les étapes suivantes :

1. La machine bootstrap démarre et commence à héberger les ressources distantes nécessaires au démarrage des machines du plan de contrôle. (Nécessite une intervention manuelle si vous approvisionnez l'infrastructure)

2. La machine d'amorçage démarre un cluster etcd à nœud unique et un plan de contrôle Kubernetes temporaire.

3. Les machines du plan de contrôle récupèrent les ressources distantes à partir de la machine d'amorçage et terminent le démarrage. (Nécessite une intervention manuelle si vous approvisionnez l'infrastructure)

4. Le plan de contrôle temporaire planifie le plan de contrôle de production sur les machines du plan de contrôle de production.

5. L'opérateur de version de cluster (CVO) se met en ligne et installe l'opérateur etcd. L'opérateur etcd met à l'échelle etcd sur tous les nœuds du plan de contrôle.

6. Le plan de contrôle temporaire s'arrête et passe le contrôle au plan de contrôle de production.

7. La machine d'amorçage injecte les composants d'OpenShift Container Platform dans le plan de contrôle de la production.

8. Le programme d'installation arrête la machine de démarrage. (Nécessite une intervention manuelle si vous approvisionnez l'infrastructure)

9. Le plan de contrôle met en place les nœuds de calcul.

10. Le plan de contrôle installe des services supplémentaires sous la forme d'un ensemble d'opérateurs.

Le résultat de ce processus de démarrage est un cluster OpenShift Container Platform en cours d'exécution. Le cluster télécharge et configure ensuite les autres composants nécessaires au fonctionnement quotidien, y compris la création de machines de calcul dans les environnements pris en charge.

Champ d'application de l'installation

Le programme d'installation d'OpenShift Container Platform est volontairement restreint. Il est conçu pour être simple et garantir le succès de l'installation. Vous pouvez effectuer de nombreuses autres tâches de configuration une fois l'installation terminée.

Ressources complémentaires

- Voir Personnalisations de cluster disponibles pour plus de détails sur les ressources de configuration d'OpenShift Container Platform.

3.2. À PROPOS DU SERVICE DE MISE À JOUR OPENSHIFT

L'OpenShift Update Service (OSUS) fournit des recommandations de mise à jour à OpenShift Container Platform, y compris Red Hat Enterprise Linux CoreOS (RHCOS). Il fournit un graphique, ou diagramme, qui contient le site *vertices* des opérateurs de composants et le site *edges* qui les relie. Les arêtes du graphique indiquent les versions que vous pouvez mettre à jour en toute sécurité. Les sommets sont des charges utiles de mise à jour qui spécifient l'état prévu des composants du cluster géré.

Le Cluster Version Operator (CVO) dans votre cluster vérifie avec l'OpenShift Update Service pour voir les mises à jour valides et les chemins de mise à jour basés sur les versions actuelles des composants et les informations dans le graphique. Lorsque vous demandez une mise à jour, le CVO utilise l'image de la version de cette mise à jour pour mettre à jour votre cluster. Les artefacts de version sont hébergés dans Quay en tant qu'images de conteneur.

Pour permettre à OpenShift Update Service de ne fournir que des mises à jour compatibles, un pipeline de vérification des versions pilote l'automatisation. Chaque artefact de version est vérifié pour sa compatibilité avec les plateformes cloud et les architectures système prises en charge, ainsi qu'avec d'autres packages de composants. Une fois que le pipeline a confirmé l'adéquation d'une version, l'OpenShift Update Service vous informe qu'elle est disponible.

IMPORTANT

Le service de mise à jour d'OpenShift affiche toutes les mises à jour recommandées pour votre cluster actuel. Si un chemin de mise à jour n'est pas recommandé par OpenShift Update Service, cela peut être dû à un problème connu avec la mise à jour ou la version cible.

Deux contrôleurs fonctionnent en mode de mise à jour continue. Le premier contrôleur met continuellement à jour les manifestes de charge utile, applique les manifestes au cluster et produit l'état de déploiement contrôlé des opérateurs pour indiquer s'ils sont disponibles, en cours de mise à niveau ou s'ils ont échoué. Le second contrôleur interroge le service de mise à jour OpenShift pour déterminer si des mises à jour sont disponibles.

IMPORTANT

Seule la mise à niveau vers une version plus récente est prise en charge. L'inversion ou le retour à une version antérieure de votre cluster n'est pas pris en charge. Si votre mise à jour échoue, contactez l'assistance Red Hat.

Au cours du processus de mise à jour, le MCO (Machine Config Operator) applique la nouvelle configuration aux machines de votre cluster. Le MCO bloque le nombre de nœuds spécifié dans le champ **maxUnavailable** du pool de configuration de la machine et les marque comme étant indisponibles. Par défaut, cette valeur est définie sur **1**. Le MCO met à jour les nœuds concernés par ordre alphabétique de zone, sur la base de l'étiquette **topology.kubernetes.io/zone**. Si une zone comporte plusieurs nœuds, les nœuds les plus anciens sont mis à jour en premier. Pour les nœuds qui

n'utilisent pas de zones, comme dans les déploiements bare metal, les nœuds sont mis à niveau par âge, les nœuds les plus anciens étant mis à jour en premier. Le MCO met à jour le nombre de nœuds spécifié par le champ **maxUnavailable** du pool de configuration de la machine à la fois. Le MCO applique ensuite la nouvelle configuration et redémarre la machine.

Si vous utilisez des machines Red Hat Enterprise Linux (RHEL) en tant que travailleurs, le MCO ne met pas à jour le kubelet car vous devez d'abord mettre à jour l'API OpenShift sur les machines.

Avec la spécification de la nouvelle version appliquée à l'ancien kubelet, la machine RHEL ne peut pas revenir à l'état **Ready**. Vous ne pouvez pas terminer la mise à jour tant que les machines ne sont pas disponibles. Toutefois, le nombre maximal de nœuds indisponibles est défini de manière à ce que les opérations normales du cluster puissent se poursuivre avec ce nombre de machines hors service.

Le service de mise à jour OpenShift est composé d'un opérateur et d'une ou plusieurs instances d'application.

3.3. POLITIQUE DE SOUTIEN AUX OPÉRATEURS NON GÉRÉS

Le site *management state* d'un opérateur détermine si celui-ci gère activement les ressources de son composant dans le cluster comme prévu. Si un opérateur est dans l'état *unmanaged*, il ne réagit pas aux changements de configuration et ne reçoit pas de mises à jour.

Bien que cela puisse être utile dans les clusters de non-production ou lors du débogage, les opérateurs dans un état non géré ne sont pas pris en charge et l'administrateur du cluster assume le contrôle total des configurations et des mises à niveau des composants individuels.

Un opérateur peut être placé dans un état non géré à l'aide des méthodes suivantes :

- **Individual Operator configuration**
  Chaque opérateur dispose d'un paramètre **managementState** dans sa configuration. Il est possible d'y accéder de différentes manières, en fonction de l'opérateur. Par exemple, l'opérateur de journalisation de Red Hat OpenShift accomplit ceci en modifiant une ressource personnalisée (CR) qu'il gère, tandis que l'opérateur d'échantillons de clusters utilise une ressource de configuration à l'échelle du cluster.

  La modification du paramètre **managementState** en **Unmanaged** signifie que l'opérateur ne gère pas activement ses ressources et qu'il ne prendra aucune mesure concernant le composant en question. Certains opérateurs peuvent ne pas supporter cet état de gestion car il pourrait endommager le cluster et nécessiter une récupération manuelle.

  AVERTISSEMENT

  Le passage d'opérateurs individuels à l'état **Unmanaged** rend ce composant et cette fonctionnalité particuliers non pris en charge. Les problèmes signalés doivent être reproduits dans l'état **Managed** pour que l'assistance soit assurée.

- **Cluster Version Operator (CVO) overrides**
  Le paramètre **spec.overrides** peut être ajouté à la configuration de l'OVC pour permettre aux administrateurs de fournir une liste de dérogations au comportement de l'OVC pour un composant. La définition du paramètre **spec.overrides[].unmanaged** à **true** pour un

composant bloque les mises à niveau du cluster et alerte l'administrateur lorsqu'une dérogation de l'OVC a été définie :

> Disabling ownership via cluster version overrides prevents upgrades. Please remove overrides before continuing.

AVERTISSEMENT

La mise en place d'une surcharge CVO place l'ensemble du cluster dans un état non supporté. Les problèmes signalés doivent être reproduits après la suppression de toute surcharge pour que l'assistance soit assurée.

3.4. PROCHAINES ÉTAPES

- Sélection d'une méthode d'installation d'un cluster et préparation pour les utilisateurs

CHAPITRE 4. RED HAT OPENSHIFT CLUSTER MANAGER

Red Hat OpenShift Cluster Manager est un service géré dans lequel vous pouvez installer, modifier, exploiter et mettre à niveau vos clusters Red Hat OpenShift. Ce service vous permet de travailler avec tous les clusters de votre organisation à partir d'un seul tableau de bord.

OpenShift Cluster Manager vous guide dans l'installation d'OpenShift Container Platform, de Red Hat OpenShift Service on AWS (ROSA) et de clusters OpenShift Dedicated. Il est également responsable de la gestion des clusters OpenShift Container Platform après l'auto-installation, ainsi que de vos clusters ROSA et OpenShift Dedicated.

Vous pouvez utiliser OpenShift Cluster Manager pour effectuer les actions suivantes :

- Créer de nouveaux clusters
- Afficher les détails et les métriques des clusters
- Gérez vos grappes en effectuant des tâches telles que la mise à l'échelle, la modification des étiquettes des nœuds, la mise en réseau et l'authentification
- Gérer le contrôle d'accès
- Contrôler les grappes d'entreprises
- Mise à jour du calendrier

4.1. ACCÈS À RED HAT OPENSHIFT CLUSTER MANAGER

Vous pouvez accéder à OpenShift Cluster Manager avec votre compte OpenShift configuré.

Conditions préalables

- Vous avez un compte qui fait partie d'une organisation OpenShift.
- Si vous créez un cluster, votre organisation a spécifié un quota.

Procédure

- Connectez-vous à OpenShift Cluster Manager Hybrid Cloud Console en utilisant vos identifiants de connexion.

4.2. ACTIONS GÉNÉRALES

En haut à droite de la page du cluster, l'utilisateur peut effectuer certaines actions sur l'ensemble du cluster :

- **Open console** lance une console web qui permet au propriétaire du cluster d'envoyer des commandes au cluster.
- **Actions** permet au propriétaire de la grappe de renommer le nom d'affichage de la grappe, de modifier la quantité d'équilibreurs de charge et de stockage persistant sur la grappe, le cas échéant, de définir manuellement le nombre de nœuds et de supprimer la grappe.
- **Refresh** force le rafraîchissement de la grappe.

4.3. ONGLETS DE LA GRAPPE

La sélection d'une grappe active et installée affiche les onglets associés à cette grappe. Les onglets suivants s'affichent une fois l'installation du cluster terminée :

- Vue d'ensemble
- Contrôle d'accès
- Compléments
- Mise en réseau
- Conseiller en matière d'information
- Pools de machines
- Soutien
- Paramètres

4.3.1. Onglet Vue d'ensemble

L'onglet **Overview** fournit des informations sur la façon dont votre cluster a été configuré :

- **Cluster ID** est l'identification unique de la grappe créée. Cet identifiant peut être utilisé lors de l'émission de commandes vers le cluster à partir de la ligne de commande.
- **Type** indique la version d'OpenShift utilisée par le cluster.
- **Region** est la région du serveur.
- **Provider** indique le fournisseur de cloud sur lequel le cluster a été construit.
- **Availability** indique le type de zone de disponibilité utilisé par le cluster, soit simple, soit multizone.
- **Version** est la version d'OpenShift installée sur le cluster. Si une mise à jour est disponible, vous pouvez la mettre à jour à partir de ce champ.
- **Created at** indique la date et l'heure de création du cluster.
- **Owner** identifie la personne qui a créé le cluster et qui dispose des droits de propriété.
- **Subscription type** indique le modèle d'abonnement sélectionné lors de la création.
- **Infrastructure type** est le type de compte utilisé par le cluster.
- **Status** affiche l'état actuel du cluster.
- **Total vCPU** indique le nombre total de CPU virtuels disponibles pour ce cluster.
- **Total memory** indique la mémoire totale disponible pour cette grappe.
- **Load balancers**
- **Persistent storage** affiche la quantité de stockage disponible sur ce cluster.

- **Nodes** indique les nœuds réels et les nœuds souhaités dans la grappe. Ces chiffres peuvent ne pas correspondre en raison de la mise à l'échelle de la grappe.
- **Network** indique l'adresse et les préfixes pour la connectivité réseau.
- **Resource usage** de l'onglet affiche les ressources en cours d'utilisation avec un graphique.
- **Advisor recommendations** donne des indications sur la sécurité, les performances, la disponibilité et la stabilité. Cette section nécessite l'utilisation de la fonctionnalité de santé à distance. Voir Utiliser Insights pour identifier les problèmes de votre cluster .
- **Cluster history** montre tout ce qui a été fait avec le cluster, y compris la création et l'identification d'une nouvelle version.

4.3.2. Onglet Contrôle d'accès

L'onglet **Access control** permet au propriétaire du cluster de configurer un fournisseur d'identité, d'accorder des autorisations élevées et d'attribuer des rôles à d'autres utilisateurs.

Conditions préalables

- Vous devez être le propriétaire de la grappe ou disposer des autorisations nécessaires pour attribuer des rôles à la grappe.

Procédure

1. Sélectionnez le bouton **Grant role**.
2. Saisissez l'identifiant du compte Red Hat de l'utilisateur auquel vous souhaitez attribuer un rôle sur le cluster.
3. Sélectionnez le bouton **Grant role** dans la boîte de dialogue.
4. La boîte de dialogue se ferme et l'utilisateur sélectionné affiche l'accès "Éditeur de cluster".

4.3.3. Onglet Add-ons

L'onglet **Add-ons** affiche tous les modules optionnels qui peuvent être ajoutés au cluster. Sélectionnez l'extension souhaitée, puis **Install** sous la description de l'extension qui s'affiche.

4.3.4. Onglet Insights Advisor

L'onglet **Insights Advisor** utilise la fonctionnalité Remote Health de OpenShift Container Platform pour identifier et atténuer les risques en matière de sécurité, de performance, de disponibilité et de stabilité. Voir Utiliser Insights pour identifier les problèmes de votre cluster dans la documentation d'OpenShift Container Platform.

4.3.5. Onglet pools de machines

L'onglet **Machine pools** permet au propriétaire du cluster de créer de nouveaux pools de machines, si le quota disponible est suffisant, ou de modifier un pool de machines existant.

La sélection de **More options** > **Scale** ouvre la boîte de dialogue "Edit node count" (Modifier le nombre de nœuds). Dans cette boîte de dialogue, vous pouvez modifier le nombre de nœuds par zone de disponibilité. Si la mise à l'échelle automatique est activée, vous pouvez également définir la plage de

mise à l'échelle automatique.

4.3.6. Onglet Support

Dans l'onglet **Support**, vous pouvez ajouter des contacts de notification pour les personnes qui doivent recevoir les notifications du cluster. Le nom d'utilisateur ou l'adresse électronique que vous fournissez doit se rapporter à un compte d'utilisateur dans l'organisation Red Hat où le cluster est déployé.

Toujours à partir de cet onglet, vous pouvez ouvrir un dossier de support pour demander une assistance technique pour votre cluster.

4.3.7. Onglet Paramètres

L'onglet **Settings** offre quelques options au propriétaire du cluster :

- **Monitoring**qui est activée par défaut, permet d'établir des rapports sur des actions définies par l'utilisateur. Voir Comprendre la pile de surveillance.
- **Update strategy** vous permet de déterminer si le cluster se met automatiquement à jour un certain jour de la semaine à une heure donnée ou si toutes les mises à jour sont planifiées manuellement.
- **Node draining** définit la durée pendant laquelle les charges de travail protégées sont respectées lors des mises à jour. Une fois cette durée écoulée, le nœud est supprimé de force.
- **Update status** indique la version actuelle et si des mises à jour sont disponibles.

4.4. RESSOURCES COMPLÉMENTAIRES

- Pour la documentation complète d'OpenShift Cluster Manager, voir la documentation d'OpenShift Cluster Manager.

CHAPITRE 5. À PROPOS DU MOTEUR MULTICLUSTER POUR L'OPÉRATEUR KUBERNETES

L'un des défis de la mise à l'échelle des environnements Kubernetes est la gestion du cycle de vie d'un parc croissant. Pour relever ce défi, vous pouvez utiliser le moteur multicluster pour l'opérateur Kubernetes (MCE). L'opérateur offre des fonctionnalités de cycle de vie complet pour les clusters OpenShift Container Platform gérés et une gestion partielle du cycle de vie pour d'autres distributions Kubernetes. Il est disponible de deux manières :

- En tant qu'opérateur autonome que vous installez dans le cadre de votre abonnement à OpenShift Container Platform ou OpenShift Kubernetes Engine
- Dans le cadre de la gestion avancée des clusters de Red Hat pour Kubernetes

5.1. GESTION DES CLUSTERS AVEC LE MOTEUR MULTICLUSTER SUR OPENSHIFT CONTAINER PLATFORM

Lorsque vous activez le moteur multicluster sur OpenShift Container Platform, vous bénéficiez des fonctionnalités suivantes :

- Les plans de contrôle hé bergés, qui est une fonctionnalité basée sur le projet HyperShift. Avec un plan de contrôle hébergé centralisé, vous pouvez exploiter des clusters OpenShift Container Platform de manière hyperscale.
- Hive, qui fournit des clusters OpenShift Container Platform autogérés au hub et complète les configurations initiales pour ces clusters.
- l'agent klusterlet, qui enregistre les clusters gérés auprès du hub.
- Infrastructure Operator, qui gère le déploiement du service assisté pour orchestrer les installations d'OpenShift Container Platform sur site en bare metal et vSphere, telles que SNO on bare metal. L'opérateur d'infrastructure inclut le zero touch provisioning (ZTP), qui automatise entièrement la création de clusters sur bare metal et le provisioning vSphere avec des workflows GitOps pour gérer les déploiements et les changements de configuration.
- Gestion des clusters ouverts, qui fournit des ressources pour gérer les clusters Kubernetes.

Le moteur multicluster est inclus dans votre abonnement de support OpenShift Container Platform et est livré séparément de la charge utile principale. Pour commencer à utiliser le moteur multicluster, vous déployez le cluster OpenShift Container Platform et installez ensuite l'opérateur. Pour plus d'informations, voir la documentation d'installation.

5.2. GESTION DE CLUSTERS AVEC RED HAT ADVANCED CLUSTER MANAGEMENT

Si vous avez besoin de capacités de gestion de cluster au-delà de ce qu'OpenShift Container Platform avec le moteur multicluster peut fournir, envisagez Red Hat Advanced Cluster Management. Le moteur multicluster fait partie intégrante de Red Hat Advanced Cluster Management et est activé par défaut.

5.3. RESSOURCES COMPLÉMENTAIRES

Pour obtenir la documentation complète du moteur multicluster, consultez la documentation du moteur multicluster, qui fait partie de la documentation du produit pour Red Hat Advanced Cluster Management.

CHAPITRE 6. ARCHITECTURE DU PLAN DE CONTRÔLE

Le site *control plane*, qui est composé de machines de plan de contrôle, gère le cluster OpenShift Container Platform. Les machines du plan de contrôle gèrent les charges de travail sur les machines de calcul, qui sont également connues sous le nom de machines de travail. Le cluster lui-même gère toutes les mises à niveau des machines grâce aux actions du Cluster Version Operator (CVO), du Machine Config Operator et d'un ensemble d'opérateurs individuels.

6.1. GESTION DE LA CONFIGURATION DES NŒUDS AVEC DES POOLS DE CONFIGURATION DE MACHINES

Les machines qui exécutent des composants du plan de contrôle ou des charges de travail utilisateur sont divisées en groupes en fonction des types de ressources qu'elles gèrent. Ces groupes de machines sont appelés pools de configuration de machines (MCP). Chaque MCP gère un ensemble de nœuds et les configurations de machines correspondantes. Le rôle du nœud détermine le MCP auquel il appartient ; le MCP régit les nœuds en fonction de l'étiquette de rôle qui lui a été attribuée. Les nœuds d'un MCP ont la même configuration, ce qui signifie que les nœuds peuvent être agrandis ou réduits en fonction de l'augmentation ou de la diminution de la charge de travail.

Par défaut, deux MCP sont créés par le cluster lors de son installation : **master** et **worker**. Chaque MCP par défaut a une configuration définie appliquée par le MCO (Machine Config Operator), qui est responsable de la gestion des MCP et de la facilitation des mises à niveau des MCP. Vous pouvez créer des MCP supplémentaires, ou des pools personnalisés, pour gérer les nœuds dont les cas d'utilisation personnalisés dépassent les types de nœuds par défaut.

Les pools personnalisés sont des pools qui héritent de la configuration du pool de travail. Ils utilisent n'importe quelle configuration de machine ciblée pour le worker pool, mais ajoutent la possibilité de déployer des changements uniquement ciblés sur le custom pool. Étant donné qu'un pool personnalisé hérite de la configuration du worker pool, toute modification apportée au worker pool est également appliquée au custom pool. Les pools personnalisés qui n'héritent pas de leur configuration du pool de travailleurs ne sont pas pris en charge par le MCO.

NOTE

Un nœud ne peut être inclus que dans un seul MCP. Si un nœud possède plusieurs étiquettes correspondant à plusieurs MCP, comme **worker,infra**, il est géré par le pool personnalisé infra, et non par le pool de travail. Les pools personnalisés sont prioritaires dans la sélection des nœuds à gérer en fonction des étiquettes des nœuds ; les nœuds qui n'appartiennent pas à un pool personnalisé sont gérés par le pool de travail.

Il est recommandé d'avoir un pool personnalisé pour chaque rôle de nœud que vous souhaitez gérer dans votre cluster. Par exemple, si vous créez des nœuds infra pour gérer des charges de travail infra, il est recommandé de créer un MCP infra personnalisé pour regrouper ces nœuds. Si vous appliquez une étiquette de rôle **infra** à un nœud de travailleur pour qu'il ait la double étiquette **worker,infra**, mais que vous n'avez pas de MCP infra personnalisé, le MCO le considère comme un nœud de travailleur. Si vous retirez l'étiquette **worker** d'un nœud et que vous appliquez l'étiquette **infra** sans le regrouper dans un pool personnalisé, le nœud n'est pas reconnu par le MCO et n'est pas géré par le cluster.

IMPORTANT

Tout nœud étiqueté avec le rôle **infra** qui exécute uniquement des charges de travail infra n'est pas pris en compte dans le nombre total d'abonnements. Le MCP qui gère un nœud infra est mutuellement exclusif de la manière dont le cluster détermine les frais d'abonnement ; l'étiquetage d'un nœud avec le rôle **infra** approprié et l'utilisation de taints pour empêcher les charges de travail utilisateur d'être planifiées sur ce nœud sont les seules conditions requises pour éviter les frais d'abonnement pour les charges de travail infra.

Le MCO applique les mises à jour des pools de manière indépendante ; par exemple, si une mise à jour affecte tous les pools, les nœuds de chaque pool sont mis à jour parallèlement les uns aux autres. Si vous ajoutez un pool personnalisé, les nœuds de ce pool tentent également de se mettre à jour simultanément avec les nœuds maître et subordonné.

Il peut arriver que la configuration d'un nœud ne corresponde pas entièrement à ce que la configuration de la machine actuellement appliquée spécifie. Cet état est appelé *configuration drift*. Le Machine Config Daemon (MCD) vérifie régulièrement que les nœuds ne présentent pas de dérive de configuration. Si le MCD détecte une dérive de la configuration, le MCO marque le nœud **degraded** jusqu'à ce qu'un administrateur corrige la configuration du nœud. Un nœud dégradé est en ligne et opérationnel, mais il ne peut pas être mis à jour.

Ressources complémentaires

- Comprendre la détection des dérives de configuration.

6.2. RÔLES DES MACHINES DANS OPENSHIFT CONTAINER PLATFORM

OpenShift Container Platform attribue aux hôtes différents rôles. Ces rôles définissent la fonction de la machine au sein du cluster. Le cluster contient des définitions pour les types de rôles standard **master** et **worker**.

NOTE

Le cluster contient également la définition du rôle **bootstrap**. La machine d'amorçage n'étant utilisée que lors de l'installation de la grappe, sa fonction est expliquée dans la documentation relative à l'installation de la grappe.

6.2.1. Compatibilité entre le plan de contrôle et l'hôte du nœud

La version d'OpenShift Container Platform doit correspondre entre l'hôte du plan de contrôle et l'hôte du nœud. Par exemple, dans un cluster 4.12, tous les hôtes du plan de contrôle doivent être 4.12 et tous les nœuds doivent être 4.12.

Les décalages temporaires lors des mises à niveau de clusters sont acceptables. Par exemple, lors de la mise à niveau d'OpenShift Container Platform 4.11 à 4.12, certains nœuds passeront à la version 4.12 avant d'autres. Un décalage prolongé des hôtes du plan de contrôle et des hôtes des nœuds peut exposer les machines de calcul plus anciennes à des bogues et à des fonctionnalités manquantes. Les utilisateurs doivent résoudre les hôtes de plan de contrôle et les hôtes de nœuds asymétriques dès que possible.

Le service **kubelet** ne doit pas être plus récent que **kube-apiserver**, et peut être jusqu'à deux versions mineures plus anciennes selon que la version de votre OpenShift Container Platform est paire ou impaire. Le tableau ci-dessous indique la compatibilité de version appropriée :

Version d'OpenShift Container Platform	Prise en charge **kubelet** skew
Versions mineures de OpenShift Container Platform [1]	Jusqu'à une version plus ancienne
Même les versions mineures de OpenShift Container Platform [2]	Jusqu'à deux versions plus anciennes

1. Par exemple, OpenShift Container Platform 4.9, 4.11.
2. Par exemple, OpenShift Container Platform 4.8, 4.10, 4.12.

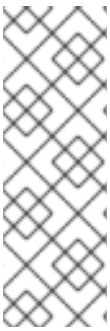
6.2.2. Travailleurs en grappe

Dans un cluster Kubernetes, les nœuds de travail sont l'endroit où les charges de travail réelles demandées par les utilisateurs de Kubernetes s'exécutent et sont gérées. Les nœuds de travail annoncent leur capacité et le planificateur, qui est un service de plan de contrôle, détermine sur quels nœuds démarrer les pods et les conteneurs. D'importants services sont exécutés sur chaque nœud de travail, notamment CRI-O, le moteur de conteneurs ; Kubelet, le service qui accepte et exécute les demandes d'exécution et d'arrêt des charges de travail des conteneurs ; un proxy de service, qui gère la communication entre les pods et les travailleurs ; et le runtime de conteneur de bas niveau runC ou crun (Technology Preview), qui crée et exécute les conteneurs.

NOTE

Pour plus d'informations sur l'activation de crun au lieu de runC par défaut, voir la documentation relative à la création d'un CR **ContainerRuntimeConfig**.

Dans OpenShift Container Platform, les ensembles de machines de calcul contrôlent les machines de calcul, auxquelles est attribué le rôle de machine **worker**. Les machines dotées du rôle **worker** pilotent des charges de travail de calcul qui sont régies par un pool de machines spécifique qui les met à l'échelle automatiquement. Comme OpenShift Container Platform a la capacité de prendre en charge plusieurs types de machines, les machines ayant le rôle **worker** sont classées comme des machines *compute*. Dans cette version, les termes *worker machine* et *compute machine* sont utilisés de manière interchangeable car le seul type de machine de calcul par défaut est la machine de travail. Dans les versions futures d'OpenShift Container Platform, d'autres types de machines de calcul, telles que les machines d'infrastructure, pourraient être utilisées par défaut.



NOTE

Les ensembles de machines de calcul sont des regroupements de ressources de machines de calcul sous l'espace de noms **machine-api**. Les ensembles de machines de calcul sont des configurations conçues pour démarrer de nouvelles machines de calcul sur un fournisseur de cloud spécifique. À l'inverse, les pools de configuration de machines (MCP) font partie de l'espace de noms Machine Config Operator (MCO). Un MCP est utilisé pour regrouper des machines afin que le MCO puisse gérer leurs configurations et faciliter leurs mises à niveau.

6.2.3. Plans de contrôle des clusters

Dans un cluster Kubernetes, les nœuds *master* exécutent les services nécessaires au contrôle du cluster Kubernetes. Dans OpenShift Container Platform, le plan de contrôle est composé de machines de plan de contrôle qui ont un rôle de machine **master**. Elles contiennent plus que les services Kubernetes pour gérer le cluster OpenShift Container Platform.

Pour la plupart des clusters OpenShift Container Platform, les machines du plan de contrôle sont définies par une série de ressources API de machines autonomes. Pour les combinaisons de fournisseurs de cloud et de versions d'OpenShift Container Platform prises en charge, les plans de contrôle peuvent être gérés avec des ensembles de machines de plan de contrôle. Des contrôles supplémentaires s'appliquent aux machines de plan de contrôle pour vous empêcher de supprimer toutes les machines de plan de contrôle et de casser votre cluster.

NOTE

Trois nœuds de plan de contrôle exactement doivent être utilisés pour tous les déploiements de production.

Les services qui relèvent de la catégorie Kubernetes sur le plan de contrôle comprennent le serveur API Kubernetes, etcd, le gestionnaire de contrôleur Kubernetes et le planificateur Kubernetes.

Tableau 6.1. Les services Kubernetes qui s'exécutent sur le plan de contrôle

Composant	Description
Serveur API Kubernetes	Le serveur API Kubernetes valide et configure les données pour les pods, les services et les contrôleurs de réplication. Il fournit également un point central pour l'état partagé du cluster.
etcd	etcd stocke l'état persistant du plan de contrôle tandis que d'autres composants surveillent etcd à la recherche de changements qui les amèneraient à l'état spécifié.
Gestionnaire de contrôleur Kubernetes	Le gestionnaire de contrôleur Kubernetes surveille 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 utilise l'API pour appliquer l'état spécifié. Plusieurs processus de ce type créent un cluster avec un leader actif à la fois.
Ordonnanceur Kubernetes	Le planificateur Kubernetes surveille les pods nouvellement créés sans nœud assigné et sélectionne le meilleur nœud pour héberger le pod.

Il existe également des services OpenShift qui s'exécutent sur le plan de contrôle, notamment le serveur API OpenShift, le gestionnaire de contrôleur OpenShift, le serveur API OpenShift OAuth et le serveur OpenShift OAuth.

Tableau 6.2. Les services OpenShift qui s'exécutent sur le plan de contrôle

Composant	Description

Composant	Description
Serveur API OpenShift	Le serveur API OpenShift valide et configure les données pour les ressources OpenShift, telles que les projets, les routes et les modèles. Le serveur OpenShift API est géré par l'opérateur OpenShift API Server.
Gestionnaire de contrôleur OpenShift	Le gestionnaire de contrôleur OpenShift surveille etcd pour les changements d'objets OpenShift, tels que les objets de projet, de route et de contrôleur de modèle, et utilise ensuite l'API pour appliquer l'état spécifié. Le gestionnaire de contrôleur OpenShift est géré par l'opérateur de gestionnaire de contrôleur OpenShift.
Serveur OpenShift OAuth API	Le serveur OpenShift OAuth API valide et configure les données d'authentification à OpenShift Container Platform, telles que les utilisateurs, les groupes et les jetons OAuth. Le serveur OpenShift OAuth API est géré par le Cluster Authentication Operator.
Serveur OpenShift OAuth	Les utilisateurs demandent des jetons au serveur OpenShift OAuth pour s'authentifier auprès de l'API. Le serveur OpenShift OAuth est géré par le Cluster Authentication Operator.

Certains de ces services sur les machines du plan de contrôle s'exécutent en tant que services systemd, tandis que d'autres s'exécutent en tant que pods statiques.

Les services Systemd sont appropriés pour les services qui doivent toujours apparaître sur un système particulier peu de temps après son démarrage. Pour les machines du plan de contrôle, il s'agit notamment de sshd, qui permet de se connecter à distance. Ils comprennent également des services tels que :

- Le moteur de conteneurs CRI-O (crio), qui exécute et gère les conteneurs. OpenShift Container Platform 4.12 utilise CRI-O à la place de Docker Container Engine.
- Kubelet (kubelet), qui accepte les demandes de gestion des conteneurs sur la machine à partir des services du plan de contrôle.

CRI-O et Kubelet doivent être exécutés directement sur l'hôte en tant que services systemd, car ils doivent fonctionner avant que vous puissiez exécuter d'autres conteneurs.

Les pods de plan de contrôle **installer-*** et **revision-pruner-*** doivent être exécutés avec les permissions de l'utilisateur root car ils écrivent dans le répertoire /**etc**/**kubernetes**, qui appartient à l'utilisateur root. Ces modules se trouvent dans les espaces de noms suivants :

- **openshift-etcd**
- **openshift-kube-apiserver**
- **openshift-kube-controller-manager**

- **openshift-kube-scheduler**

6.3. OPÉRATEURS DANS OPENSHIFT CONTAINER PLATFORM

Les opérateurs sont parmi les composants les plus importants d'OpenShift Container Platform. Les opérateurs sont la méthode privilégiée pour conditionner, déployer et gérer les services sur le plan de contrôle. Ils peuvent également offrir des avantages aux applications exécutées par les utilisateurs.

Les opérateurs s'intègrent aux API de Kubernetes et aux outils CLI tels que les commandes **kubectl** et **oc**. Ils permettent de surveiller les applications, d'effectuer des contrôles de santé, de gérer les mises à jour over-the-air (OTA) et de s'assurer que les applications restent dans l'état que vous avez spécifié.

Les opérateurs offrent également une expérience de configuration plus granulaire. Vous configurez chaque composant en modifiant l'API que l'opérateur expose au lieu de modifier un fichier de configuration global.

CRI-O et Kubelet s'exécutant sur chaque nœud, presque toutes les autres fonctions du cluster peuvent être gérées sur le plan de contrôle à l'aide d'opérateurs. Les composants ajoutés au plan de contrôle à l'aide d'opérateurs comprennent les services critiques de réseau et d'authentification.

Bien que les deux suivent des concepts et des objectifs similaires pour les opérateurs, les opérateurs dans OpenShift Container Platform sont gérés par deux systèmes différents, en fonction de leur objectif :

- Les opérateurs de cluster, qui sont gérés par l'opérateur de version de cluster (CVO), sont installés par défaut pour exécuter les fonctions de cluster.
- Des opérateurs supplémentaires facultatifs, gérés par Operator Lifecycle Manager (OLM), peuvent être mis à la disposition des utilisateurs pour qu'ils les exécutent dans leurs applications.

6.3.1. Opérateurs de groupe

Dans OpenShift Container Platform, toutes les fonctions de cluster sont divisées en une série de *cluster Operators* par défaut. Les opérateurs de cluster gèrent un domaine particulier des fonctionnalités du cluster, comme la journalisation des applications à l'échelle du cluster, la gestion du plan de contrôle Kubernetes ou le système de provisionnement des machines.

Les opérateurs de cluster sont représentés par un objet **ClusterOperator**, que les administrateurs de cluster peuvent consulter dans la console web d'OpenShift Container Platform à partir de la page **Administration → Cluster Settings**. Chaque opérateur de cluster fournit une API simple pour déterminer les fonctionnalités du cluster. L'Opérateur cache les détails de la gestion du cycle de vie de ce composant. Les opérateurs peuvent gérer un seul composant ou des dizaines de composants, mais l'objectif final est toujours de réduire la charge opérationnelle en automatisant les actions courantes.

Ressources complémentaires

- Référence des opérateurs de clusters

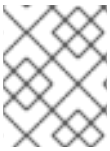
6.3.2. Opérateurs complémentaires

Operator Lifecycle Manager (OLM) et OperatorHub sont des composants par défaut dans OpenShift Container Platform qui aident à gérer les applications natives de Kubernetes en tant qu'opérateurs. Ensemble, ils fournissent le système de découverte, d'installation et de gestion des opérateurs complémentaires optionnels disponibles sur le cluster.

En utilisant OperatorHub dans la console web d'OpenShift Container Platform, les administrateurs de clusters et les utilisateurs autorisés peuvent sélectionner des opérateurs à installer à partir de catalogues d'opérateurs. Après avoir installé un opérateur à partir d'OperatorHub, il peut être mis à disposition globalement ou dans des espaces de noms spécifiques pour être exécuté dans les applications des utilisateurs.

Des sources de catalogue par défaut sont disponibles et incluent les opérateurs Red Hat, les opérateurs certifiés et les opérateurs communautaires. Les administrateurs de clusters peuvent également ajouter leurs propres sources de catalogue personnalisées, qui peuvent contenir un ensemble personnalisé d'opérateurs.

Les développeurs peuvent utiliser le SDK de l'opérateur pour créer des opérateurs personnalisés qui tirent parti des fonctionnalités d'OLM. Leur opérateur peut ensuite être regroupé et ajouté à un catalogue source personnalisé, qui peut être ajouté à un cluster et mis à la disposition des utilisateurs.



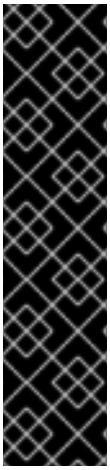
NOTE

OLM ne gère pas les opérateurs de clusters qui constituent l'architecture d'OpenShift Container Platform.

Ressources complémentaires

- Pour plus de détails sur l'exécution d'opérateurs complémentaires dans OpenShift Container Platform, voir les sections du guide *Operators* sur Operator Lifecycle Manager (OLM) et OperatorHub.
- Pour plus de détails sur le SDK de l'opérateur, voir Développement d'opérateurs.

6.3.3. Opérateurs de plateforme (aperçu technologique)



IMPORTANT

Le type d'opérateur de plate-forme est une fonctionnalité d'aperçu technologique uniquement. Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat et peuvent ne pas être complètes sur le plan fonctionnel. Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Pour plus d'informations sur la portée de l'assistance des fonctionnalités de l'aperçu technologique de Red Hat, voir Portée de l'assistance des fonctionnalités de l'aperçu technologique.

Operator Lifecycle Manager (OLM) introduit un nouveau type d'opérateur appelé *platform Operators*. Un opérateur de plateforme est un opérateur basé sur OLM qui peut être installé pendant ou après les opérations du jour 0 d'un cluster OpenShift Container Platform et qui participe au cycle de vie du cluster. En tant qu'administrateur de cluster, vous pouvez utiliser les opérateurs de plateforme pour personnaliser davantage votre installation OpenShift Container Platform afin de répondre à vos exigences et à vos cas d'utilisation.

En utilisant la fonction existante de capacités de cluster dans OpenShift Container Platform, les administrateurs de cluster peuvent déjà désactiver un sous-ensemble de composants basés sur l'opérateur de version de cluster (CVO) considérés comme non essentiels à la charge utile initiale avant l'installation du cluster. Les opérateurs de plateforme s'inspirent de ce modèle en offrant des options de

personnalisation supplémentaires. Grâce au mécanisme de l'opérateur de plateforme, qui s'appuie sur les ressources du composant RukPak, les opérateurs basés sur OLM peuvent désormais être installés au moment de l'installation du cluster et peuvent bloquer le déploiement du cluster si l'opérateur ne parvient pas à s'installer correctement.

Dans OpenShift Container Platform 4.12, cette version Technology Preview se concentre sur le mécanisme de base de la plateforme Operator et construit une base pour étendre le concept dans les prochaines versions. Vous pouvez utiliser l'API **PlatformOperator** à l'échelle du cluster pour configurer les opérateurs avant ou après la création du cluster sur les clusters qui ont activé l'ensemble de fonctionnalités **TechPreviewNoUpgrades**.

Ressources complémentaires

- Gestion des opérateurs de plateforme
- Restrictions relatives à l'aperçu technologique pour les opérateurs de plates-formes
- Composants et format d'emballage RukPak
- Capacités des clusters

6.4. À PROPOS DE L'OPÉRATEUR DE CONFIGURATION DE LA MACHINE

OpenShift Container Platform 4.12 intègre à la fois la gestion du système d'exploitation et du cluster. Étant donné que le cluster gère ses propres mises à jour, y compris les mises à jour de Red Hat Enterprise Linux CoreOS (RHCOS) sur les nœuds du cluster, OpenShift Container Platform offre une expérience de gestion du cycle de vie basée sur l'opinion qui simplifie l'orchestration des mises à niveau des nœuds.

OpenShift Container Platform utilise trois ensembles de démons et contrôleurs pour simplifier la gestion des nœuds. Ces ensembles de démons orchestrent les mises à jour du système d'exploitation et les modifications de configuration des hôtes en utilisant des constructions standard de type Kubernetes. Il s'agit de :

- Le site **machine-config-controller**, qui coordonne les mises à niveau des machines à partir du plan de contrôle. Il surveille tous les nœuds de la grappe et orchestre leurs mises à jour de configuration.
- L'ensemble de démons **machine-config-daemon**, qui s'exécute sur chaque nœud du cluster et met à jour la configuration d'une machine telle que définie par machine config et selon les instructions du MachineConfigController. Lorsque le nœud détecte un changement, il vide ses pods, applique la mise à jour et redémarre. Ces changements se présentent sous la forme de fichiers de configuration Ignition qui appliquent la configuration machine spécifiée et contrôlent la configuration des kubelets. La mise à jour elle-même est livrée dans un conteneur. Ce processus est la clé du succès de la gestion conjointe des mises à jour d'OpenShift Container Platform et de RHCOS.
- Le jeu de démons **machine-config-server**, qui fournit les fichiers de configuration Ignition pour contrôler les nœuds de l'avion lorsqu'ils rejoignent le cluster.

La configuration de la machine est un sous-ensemble de la configuration d'Ignition. Le site **machine-config-daemon** lit la configuration de la machine pour voir s'il doit effectuer une mise à jour d'OSTree ou s'il doit appliquer une série de modifications de fichiers kubelet systemd, de modifications de configuration ou d'autres modifications du système d'exploitation ou de la configuration d'OpenShift Container Platform.

Lorsque vous effectuez des opérations de gestion de nœuds, vous créez ou modifiez une ressource personnalisée (CR) **KubeletConfig**.

IMPORTANT

Lorsque des modifications sont apportées à la configuration d'une machine, le MCO (Machine Config Operator) redémarre automatiquement tous les nœuds correspondants pour que les modifications soient prises en compte.

Pour éviter que les nœuds ne redémarrent automatiquement après des modifications de la configuration de la machine, vous devez, avant d'effectuer ces modifications, interrompre le processus de redémarrage automatique en définissant le champ **spec.paused** sur **true** dans le pool de configuration de la machine correspondant. En cas de pause, les modifications de la configuration de la machine ne sont pas appliquées tant que vous n'avez pas défini le champ **spec.paused** sur **false** et que les nœuds n'ont pas redémarré dans la nouvelle configuration.

Assurez-vous que les pools ne sont pas interrompus lorsque la rotation des certificats d'AC a lieu. Si les MCP sont en pause, le MCO ne peut pas envoyer les nouveaux certificats à ces nœuds. Cela entraîne la dégradation du cluster et l'échec de plusieurs commandes **oc**, notamment **oc debug**, **oc logs**, **oc exec** et **oc attach**. Vous recevez des alertes dans l'interface utilisateur d'alerte de la console Web d'OpenShift Container Platform si un MCP est mis en pause lors de la rotation des certificats.

Les modifications suivantes ne déclenchent pas de redémarrage du nœud :

- Lorsque le MCO détecte l'un des changements suivants, il applique la mise à jour sans vidanger ou redémarrer le nœud :
  - Modifications de la clé SSH dans le paramètre **spec.config.passwd.users.sshAuthorizedKeys** de la configuration d'une machine.
  - Changements apportés au secret de tirage global ou au secret de tirage dans l'espace de noms **openshift-config**.
  - Rotation automatique de l'autorité de certification **/etc/kubernetes/kubelet-ca.crt** par l'opérateur du serveur API Kubernetes.
- Lorsque le MCO détecte des modifications dans le fichier **/etc/containers/registries.conf**, telles que l'ajout ou la modification d'un objet **ImageDigestMirrorSet** ou **ImageTagMirrorSet**, il draine les nœuds correspondants, applique les modifications et désenregistre les nœuds :
  - L'ajout d'un registre avec le jeu de paramètres **pull-from-mirror = "digest-only"** pour chaque miroir.
  - L'ajout d'un miroir avec le paramètre **pull-from-mirror = "digest-only"** défini dans un registre.
  - L'ajout d'éléments à la liste **unqualified-search-registries**.

Il peut arriver que la configuration d'un nœud ne corresponde pas entièrement à ce que la configuration de la machine actuellement appliquée spécifie. Cet état est appelé *configuration drift*. Le Machine Config Daemon (MCD) vérifie régulièrement que les nœuds ne présentent pas de dérive de

configuration. Si le MCD détecte une dérive de la configuration, le MCO marque le nœud **degraded** jusqu'à ce qu'un administrateur corrige la configuration du nœud. Un nœud dégradé est en ligne et opérationnel, mais il ne peut pas être mis à jour.

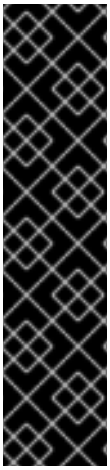
Ressources complémentaires

- Pour plus d'informations sur la détection de la dérive de la configuration, voir Comprendre la détection de la dérive de la configuration.
- Pour plus d'informations sur la façon d'empêcher les machines du plan de contrôle de redémarrer après que l'opérateur de configuration des machines a apporté des modifications à la configuration des machines, voir Désactiver le redémarrage automatique de l'opérateur de configuration des machines.

6.5. APERÇU DES PLANS DE CONTRÔLE HÉBERGÉS (APERÇU TECHNOLOGIQUE)

Vous pouvez utiliser des plans de contrôle hébergés pour Red Hat OpenShift Container Platform afin de réduire les coûts de gestion, d'optimiser le temps de déploiement des clusters et de séparer les préoccupations liées à la gestion et à la charge de travail afin que vous puissiez vous concentrer sur vos applications.

Vous pouvez activer les plans de contrôle hébergés en tant que fonctionnalité d'aperçu technologique en utilisant le moteur multicluster pour l'opérateur Kubernetes version 2.0 ou ultérieure sur Amazon Web Services (AWS).



IMPORTANT

Les plans de contrôle hébergés sont une fonctionnalité d'aperçu technologique uniquement. Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat et peuvent ne pas être complètes sur le plan fonctionnel. Red Hat ne recommande pas de les utiliser en production. Ces fonctionnalités offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Pour plus d'informations sur la portée de l'assistance des fonctionnalités de l'aperçu technologique de Red Hat, voir Portée de l'assistance des fonctionnalités de l'aperçu technologique.

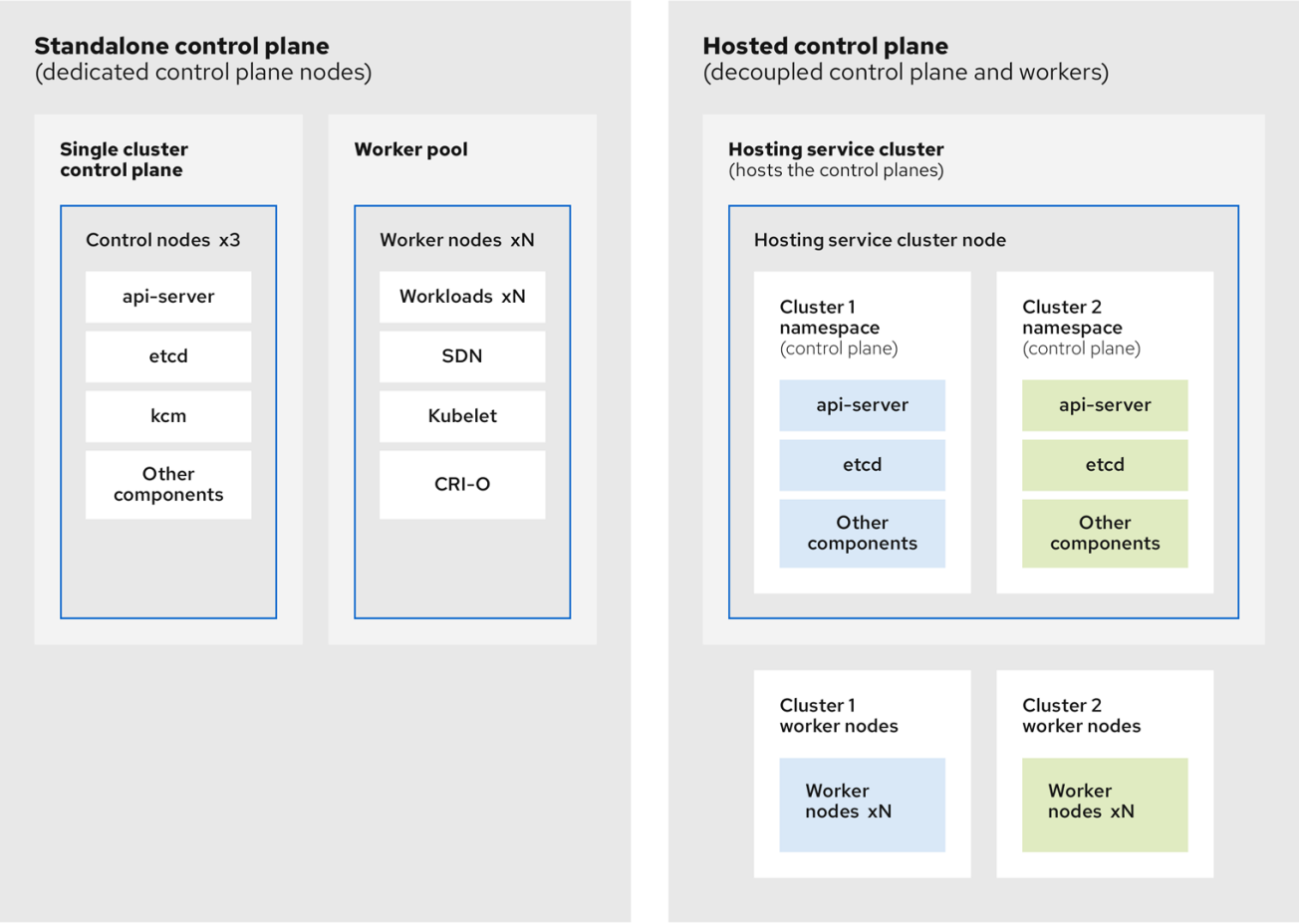
6.5.1. Architecture des plans de contrôle hébergés

OpenShift Container Platform est souvent déployé dans un modèle couplé, ou autonome, où un cluster se compose d'un plan de contrôle et d'un plan de données. Le plan de contrôle comprend un point d'extrémité d'API, un point d'extrémité de stockage, un planificateur de charge de travail et un actionneur qui assure l'état. Le plan de données comprend le calcul, le stockage et le réseau où s'exécutent les charges de travail et les applications.

Le plan de contrôle autonome est hébergé par un groupe dédié de nœuds, qui peuvent être physiques ou virtuels, avec un nombre minimum pour assurer le quorum. La pile réseau est partagée. L'accès de l'administrateur à une grappe offre une visibilité sur le plan de contrôle de la grappe, les API de gestion des machines et d'autres composants qui contribuent à l'état d'une grappe.

Bien que le modèle autonome fonctionne bien, certaines situations exigent une architecture où le plan de contrôle et le plan de données sont découplés. Dans ce cas, le plan de données se trouve sur un

domaine de réseau séparé avec un environnement d'hébergement physique dédié. Le plan de contrôle est hébergé en utilisant des primitives de haut niveau telles que les déploiements et les ensembles avec état qui sont natifs de Kubernetes. Le plan de contrôle est traité comme n'importe quelle autre charge de travail.



6.5.2. Avantages des plans de contrôle hébergés

Avec les plans de contrôle hébergés pour OpenShift Container Platform, vous pouvez ouvrir la voie à une véritable approche de cloud hybride et profiter de plusieurs autres avantages.

- Les frontières de sécurité entre la gestion et les charges de travail sont plus solides car le plan de contrôle est découplé et hébergé sur un cluster de service d'hébergement dédié. Par conséquent, vous êtes moins susceptible de divulguer les informations d'identification des clusters à d'autres utilisateurs. La gestion des comptes secrets de l'infrastructure étant également découplée, les administrateurs de l'infrastructure des clusters ne peuvent pas supprimer accidentellement l'infrastructure du plan de contrôle.

- Avec les plans de contrôle hébergés, vous pouvez exécuter de nombreux plans de contrôle sur un nombre réduit de nœuds. Les grappes sont donc plus abordables.

- Comme les plans de contrôle sont constitués de pods lancés sur OpenShift Container Platform, les plans de contrôle démarrent rapidement. Les mêmes principes s'appliquent aux plans de contrôle et aux charges de travail, telles que la surveillance, la journalisation et la mise à l'échelle automatique.

- Du point de vue de l'infrastructure, vous pouvez pousser les registres, HAProxy, la surveillance des clusters, les nœuds de stockage et d'autres composants de l'infrastructure vers le compte du fournisseur de cloud du locataire, isolant ainsi l'utilisation du locataire.

- D'un point de vue opérationnel, la gestion des grappes multiples est plus centralisée, ce qui réduit le nombre de facteurs externes qui affectent l'état et la cohérence de la grappe. Les ingénieurs de fiabilité du site disposent d'un endroit central pour déboguer les problèmes et naviguer vers le plan de données du cluster, ce qui peut réduire le temps de résolution (TTR) et augmenter la productivité.

Ressources complémentaires

- Module complémentaire HyperShift (aperçu technologique)
- Plans de contrôle hébergés pour Red Hat OpenShift Container Platform (aperçu technologique)

6.5.3. Versioning pour les plans de contrôle hébergés

À chaque version majeure, mineure ou corrective d'OpenShift Container Platform, deux composants des plans de contrôle hébergés sont publiés :

- Opérateur HyperShift
- Interface de ligne de commande (CLI)

L'opérateur HyperShift gère le cycle de vie des clusters hébergés qui sont représentés par les ressources de l'API **HostedCluster**. L'opérateur HyperShift est publié avec chaque version d'OpenShift Container Platform.

Le CLI est un utilitaire d'aide au développement. Le CLI fait partie de toute version d'HyperShift Operator. Aucune politique de compatibilité n'est garantie.

L'API, **hypershift.openshift.io**, permet de créer et de gérer des clusters OpenShift Container Platform légers, flexibles et hétérogènes à l'échelle. L'API expose deux ressources orientées utilisateur : **HostedCluster** et **NodePool**. Une ressource **HostedCluster** encapsule le plan de contrôle et la configuration commune du plan de données. Lorsque vous créez une ressource **HostedCluster**, vous disposez d'un plan de contrôle entièrement fonctionnel sans nœuds attachés. Une ressource **NodePool** est un ensemble évolutif de nœuds de travail attaché à une ressource **HostedCluster**.

La politique de version de l'API s'aligne généralement sur la politique de version de l'API Kubernetes .

Ressources complémentaires

- Pour plus d'informations sur les ressources **HostedCluster** et **NodePool**, voir la référence API HyperShift.

CHAPITRE 7. COMPRENDRE LE DÉVELOPPEMENT DE LA PLATEFORME DE CONTENEURS OPENSHIFT

Pour tirer pleinement parti des capacités des conteneurs lors du développement et de l'exécution d'applications de qualité professionnelle, assurez-vous que votre environnement est pris en charge par des outils qui permettent aux conteneurs d'être utilisés :

- Créés en tant que microservices distincts pouvant être connectés à d'autres services conteneurisés et non conteneurisés. Par exemple, vous pourriez vouloir relier votre application à une base de données ou y attacher une application de surveillance.
- Résilience : si un serveur tombe en panne ou doit être mis hors service pour des raisons de maintenance ou de démantèlement, les conteneurs peuvent démarrer sur une autre machine.
- Automatisé pour récupérer automatiquement les changements de code, puis démarrer et déployer de nouvelles versions d'eux-mêmes.
- Mise à l'échelle, ou réplication, pour que davantage d'instances servent les clients lorsque la demande augmente, puis réduction du nombre d'instances en cas de baisse de la demande.
- S'exécuter de différentes manières, selon le type d'application. Par exemple, une application peut être exécutée une fois par mois pour produire un rapport et se terminer ensuite. Une autre application peut avoir besoin de fonctionner en permanence et d'être hautement disponible pour les clients.
- Géré de manière à ce que vous puissiez surveiller l'état de votre application et réagir en cas de problème.

L'acceptation généralisée des conteneurs et les exigences qui en découlent en matière d'outils et de méthodes pour les adapter aux besoins de l'entreprise ont donné lieu à de nombreuses options.

Le reste de cette section explique les options pour les actifs que vous pouvez créer lorsque vous construisez et déployez des applications Kubernetes conteneurisées dans OpenShift Container Platform. Elle décrit également les approches que vous pouvez utiliser pour différents types d'applications et d'exigences de développement.

7.1. A PROPOS DU DÉVELOPPEMENT D'APPLICATIONS CONTENEURISÉES

Vous pouvez aborder le développement d'applications avec des conteneurs de nombreuses façons, et différentes approches peuvent être plus appropriées pour différentes situations. Pour illustrer cette diversité, la série d'approches présentée commence par le développement d'un seul conteneur et finit par le déploiement de ce conteneur en tant qu'application critique pour une grande entreprise. Ces approches montrent les différents outils, formats et méthodes que vous pouvez utiliser pour le développement d'applications conteneurisées. Cette rubrique décrit :

- Construire un conteneur simple et le stocker dans un registre
- Créer un manifeste Kubernetes et l'enregistrer dans un dépôt Git
- Créer un opérateur pour partager votre candidature avec d'autres personnes

7.2. CONSTRUCTION D'UN CONTENEUR SIMPLE

Vous avez une idée d'application et vous voulez la conteneuriser.

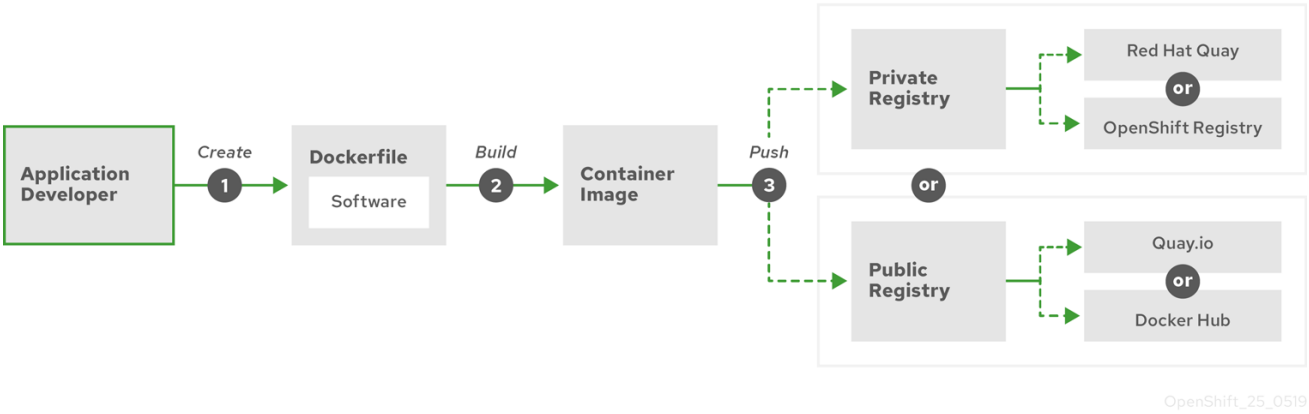
Tout d'abord, vous avez besoin d'un outil pour construire un conteneur, comme buildah ou docker, et d'un fichier qui décrit ce qui va dans votre conteneur, qui est typiquement un Dockerfile.

Ensuite, vous avez besoin d'un emplacement pour pousser l'image de conteneur résultante afin que vous puissiez l'extraire pour qu'elle s'exécute là où vous le souhaitez. Cet emplacement est un registre de conteneurs.

Certains exemples de chacun de ces composants sont installés par défaut sur la plupart des systèmes d'exploitation Linux, à l'exception du fichier Dockerfile, que vous fournissez vous-même.

Le diagramme suivant illustre le processus de construction et de poussée d'une image :

Figure 7.1. Créer une application conteneurisée simple et la pousser vers un registre



Si vous utilisez un ordinateur dont le système d'exploitation est Red Hat Enterprise Linux (RHEL), le processus de création d'une application conteneurisée nécessite les étapes suivantes :

1. Installer les outils de construction de conteneurs : RHEL contient un ensemble d'outils, dont podman, buildah et skopeo, que vous utilisez pour construire et gérer des conteneurs.

2. Créer un fichier Docker pour combiner l'image de base et les logiciels : Les informations relatives à la construction de votre conteneur sont consignées dans un fichier nommé **Dockerfile**. Dans ce fichier, vous identifiez l'image de base à partir de laquelle vous construisez votre conteneur, les logiciels que vous installez et ceux que vous copiez dans le conteneur. Vous identifiez également les valeurs des paramètres tels que les ports réseau que vous exposez à l'extérieur du conteneur et les volumes que vous montez à l'intérieur du conteneur. Placez votre fichier Docker et le logiciel que vous souhaitez conteneuriser dans un répertoire de votre système RHEL.

3. Exécutez buildah ou docker build : Exécutez la commande **buildah build-using-dockerfile** ou **docker build** pour extraire l'image de base que vous avez choisie vers le système local et créer une image de conteneur qui est stockée localement. Vous pouvez également créer des images de conteneurs sans fichier Docker en utilisant buildah.

4. Marquer et pousser vers un registre : Ajoutez une balise à votre nouvelle image de conteneur qui identifie l'emplacement du registre dans lequel vous souhaitez stocker et partager votre conteneur. Poussez ensuite cette image vers le registre en exécutant la commande **podman push** ou **docker push**.

5. Extrayez et exécutez l'image : À partir de n'importe quel système doté d'un outil client de conteneur, tel que podman ou docker, exécutez une commande qui identifie votre nouvelle

image. Par exemple, exécutez la commande **podman run <image_name>** ou **docker run <image_name>**. Ici, **<image_name>** est le nom de votre nouvelle image de conteneur, qui ressemble à **quay.io/myrepo/myapp:latest**. Le registre peut exiger des informations d'identification pour pousser et tirer des images.

Pour plus de détails sur le processus de construction d'images de conteneurs, leur transfert dans les registres et leur exécution, voir Constructions d'images personnalisées avec Buildah .

7.2.1. Options de l'outil de construction de conteneurs

La construction et la gestion de conteneurs avec buildah, podman et skopeo permettent d'obtenir des images de conteneurs standard qui incluent des fonctionnalités spécifiquement adaptées au déploiement de conteneurs dans OpenShift Container Platform ou d'autres environnements Kubernetes. Ces outils sont sans démon et peuvent s'exécuter sans privilèges root, ce qui nécessite moins de frais généraux pour les faire fonctionner.

IMPORTANT

La prise en charge de Docker Container Engine en tant que moteur d'exécution de conteneur est obsolète dans Kubernetes 1.20 et sera supprimée dans une prochaine version. Cependant, les images produites par Docker continueront à fonctionner dans votre cluster avec tous les runtimes, y compris CRI-O. Pour plus d'informations, consultez l'annonce du blog Kubernetes .

Lorsque vous exécutez vos conteneurs dans OpenShift Container Platform, vous utilisez le moteur de conteneur CRI-O. CRI-O s'exécute sur chaque machine worker et control plane dans un cluster OpenShift Container Platform, mais CRI-O n'est pas encore supporté en tant que runtime autonome en dehors d'OpenShift Container Platform.

7.2.2. Options de l'image de base

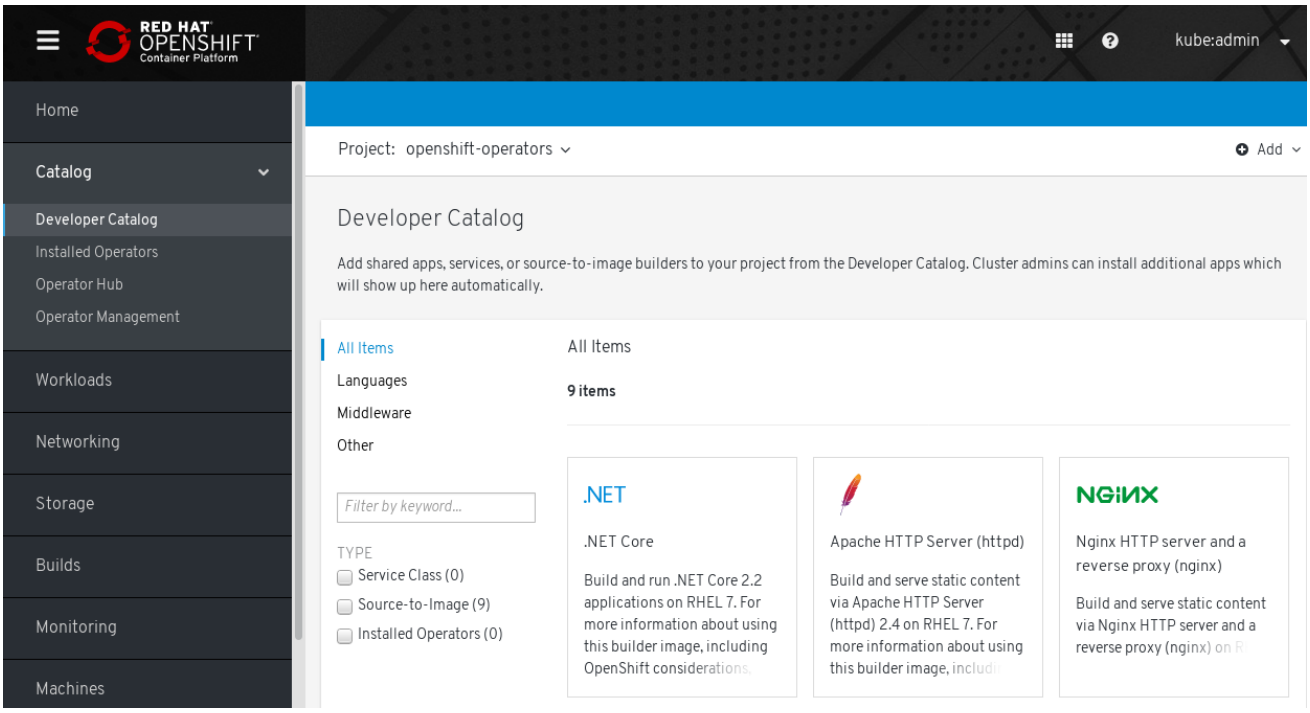
L'image de base sur laquelle vous choisissez de construire votre application contient un ensemble de logiciels qui ressemble à un système Linux pour votre application. Lorsque vous créez votre propre image, votre logiciel est placé dans ce système de fichiers et voit ce système de fichiers comme s'il regardait son système d'exploitation. Le choix de cette image de base a un impact majeur sur la sécurité, l'efficacité et l'évolutivité de votre conteneur à l'avenir.

Red Hat propose un nouvel ensemble d'images de base appelées Red Hat Universal Base Images (UBI). Ces images sont basées sur Red Hat Enterprise Linux et sont similaires aux images de base que Red Hat a proposées par le passé, avec une différence majeure : elles sont librement redistribuables sans abonnement à Red Hat. Par conséquent, vous pouvez développer votre application sur des images UBI sans avoir à vous soucier de la manière dont elles sont partagées ou de la nécessité de créer des images différentes pour des environnements différents.

Ces images UBI ont des versions standard, init et minimale. Vous pouvez également utiliser les images de Red Hat Software Collections comme base pour les applications qui reposent sur des environnements d'exécution spécifiques tels que Node.js, Perl ou Python. Des versions spéciales de certaines de ces images de base d'exécution sont appelées images Source-to-Image (S2I). Avec les images S2I, vous pouvez insérer votre code dans un environnement d'image de base prêt à l'exécuter.

Les images S2I sont disponibles pour être utilisées directement depuis l'interface web d'OpenShift Container Platform en sélectionnant **Catalog → Developer Catalog**, comme le montre la figure suivante :

Figure 7.2. Choisir des images de base S2I pour les applications nécessitant des runtimes spécifiques



7.2.3. Options du registre

Les registres de conteneurs sont l'endroit où vous stockez les images de conteneurs afin de les partager avec d'autres et de les mettre à la disposition de la plateforme sur laquelle elles sont exécutées. Vous pouvez sélectionner de grands registres de conteneurs publics qui proposent des comptes gratuits ou une version premium offrant davantage de stockage et de fonctionnalités spéciales. Vous pouvez également installer votre propre registre qui peut être exclusif à votre organisation ou partagé sélectivement avec d'autres.

Pour obtenir des images Red Hat et des images de partenaires certifiés, vous pouvez puiser dans le Red Hat Registry. Le Red Hat Registry est représenté par deux emplacements : **registry.access.redhat.com** red Hat Registry, qui n'est pas authentifié et qui est obsolète, et **registry.redhat.io**, qui nécessite une authentification. Vous pouvez en savoir plus sur les images Red Hat et les images des partenaires dans le Red Hat Registry à partir de la section Images de conteneurs du catalogue de l'écosystème Red Hat . En plus de répertorier les images de conteneurs Red Hat, le catalogue présente également des informations détaillées sur le contenu et la qualité de ces images, y compris des scores de santé basés sur les mises à jour de sécurité appliquées.

Les grands registres publics comprennent Docker Hub et Quay.io. Le registre Quay.io est détenu et géré par Red Hat. De nombreux composants utilisés dans OpenShift Container Platform sont stockés dans Quay.io, notamment les images de conteneurs et les opérateurs utilisés pour déployer OpenShift Container Platform elle-même. Quay.io permet également de stocker d'autres types de contenu, notamment les graphiques Helm.

Si vous souhaitez disposer de votre propre registre de conteneurs, OpenShift Container Platform inclut un registre de conteneurs privé qui est installé avec OpenShift Container Platform et fonctionne sur son cluster. Red Hat propose également une version privée du registre Quay.io appelée Red Hat Quay. Red Hat Quay comprend la réplication géographique, les déclencheurs de construction Git, l'analyse d'images Clair et de nombreuses autres fonctionnalités.

Tous les registres mentionnés ici peuvent exiger des informations d'identification pour télécharger des images à partir de ces registres. Certains de ces identifiants sont présentés à l'échelle du cluster par OpenShift Container Platform, tandis que d'autres peuvent être attribués à des individus.

7.3. CRÉER UN MANIFESTE KUBERNETES POUR OPENSHIFT CONTAINER PLATFORM

Bien que l'image de conteneur soit l'élément de base d'une application conteneurisée, davantage d'informations sont nécessaires pour gérer et déployer cette application dans un environnement Kubernetes tel qu'OpenShift Container Platform. Après la création d'une image, les étapes suivantes sont généralement les suivantes :

- Comprendre les différentes ressources avec lesquelles vous travaillez dans les manifestes Kubernetes
- Prenez des décisions concernant le type d'application que vous utilisez
- Rassembler les éléments de soutien
- Créez un manifeste et stockez-le dans un dépôt Git afin de pouvoir le stocker dans un système de versionnement des sources, l'auditer, le suivre, le promouvoir et le déployer dans l'environnement suivant, revenir à des versions antérieures, si nécessaire, et le partager avec d'autres

7.3.1. À propos des pods et des services Kubernetes

Alors que l'image du conteneur est l'unité de base de Docker, les unités de base avec lesquelles Kubernetes travaille sont appelées pods. Les pods représentent l'étape suivante dans la construction d'une application. Un pod peut contenir un ou plusieurs conteneurs. L'essentiel est que le pod soit l'unité unique que vous déployez, mettez à l'échelle et gérez.

L'évolutivité et les espaces de noms sont probablement les principaux éléments à prendre en compte pour déterminer ce qu'il faut mettre dans un pod. Pour faciliter le déploiement, vous pouvez vouloir déployer un conteneur dans un pod et y inclure son propre conteneur de journalisation et de surveillance. Plus tard, lorsque vous ferez fonctionner le pod et que vous aurez besoin de mettre à l'échelle une instance supplémentaire, ces autres conteneurs seront mis à l'échelle en même temps que lui. Pour les espaces de noms, les conteneurs d'un module partagent les mêmes interfaces réseau, les mêmes volumes de stockage partagés et les mêmes limites de ressources, telles que la mémoire et l'unité centrale, ce qui facilite la gestion du contenu du module en tant qu'unité unique. Les conteneurs d'un pod peuvent également communiquer entre eux en utilisant des communications inter-processus standard, telles que les sémaphores System V ou la mémoire partagée POSIX.

Alors que les pods individuels représentent une unité évolutive dans Kubernetes, un service permet de regrouper un ensemble de pods pour créer une application complète et stable capable d'effectuer des tâches telles que l'équilibrage de charge. Un service est également plus permanent qu'un pod, car il reste disponible à partir de la même adresse IP jusqu'à ce que vous le supprimiez. Lorsque le service est utilisé, il est demandé par son nom et le cluster OpenShift Container Platform résout ce nom en adresses IP et ports permettant d'atteindre les pods qui composent le service.

Par nature, les applications conteneurisées sont séparées des systèmes d'exploitation sur lesquels elles s'exécutent et, par extension, de leurs utilisateurs. Une partie de votre manifeste Kubernetes décrit comment exposer l'application aux réseaux internes et externes en définissant des stratégies réseau qui permettent un contrôle fin de la communication avec vos applications conteneurisées. Pour connecter des requêtes entrantes pour HTTP, HTTPS et d'autres services provenant de l'extérieur de votre cluster à des services à l'intérieur de votre cluster, vous pouvez utiliser une ressource **Ingress** ressource.

Si votre conteneur nécessite un stockage sur disque au lieu d'un stockage de base de données, qui peut être fourni par un service, vous pouvez ajouter des volumes à vos manifestes pour mettre ce stockage à la disposition de vos pods. Vous pouvez configurer les manifestes pour créer des volumes persistants (PV) ou créer dynamiquement des volumes qui sont ajoutés à vos définitions **Pod**.

Après avoir défini un groupe de pods qui composent votre application, vous pouvez définir ces pods dans **Deployment** et **DeploymentConfig** objets.

7.3.2. Types d'applications

Ensuite, réfléchissez à la manière dont votre type d'application influe sur la façon de l'exécuter.

Kubernetes définit différents types de charges de travail qui conviennent à différents types d'applications. Pour déterminer la charge de travail appropriée à votre application, déterminez si l'application est.. :

- Il est destiné à être exécuté jusqu'à son terme et à être terminé. Par exemple, une application qui démarre pour produire un rapport et qui se termine lorsque le rapport est terminé. Il se peut que l'application ne s'exécute plus pendant un mois. Les objets OpenShift Container Platform adaptés à ce type d'applications sont les suivants **Job** et **CronJob** objets.

- Ces applications sont censées fonctionner en continu. Pour les applications à longue durée d'exécution, vous pouvez écrire un déploiement.

- Nécessaire pour être hautement disponible. Si votre application nécessite une haute disponibilité, vous devez dimensionner votre déploiement de manière à avoir plus d'une instance. Un objet **Deployment** ou **DeploymentConfig** peut intégrer un ensemble de répliques pour ce type d'application. Avec les ensembles de répliques, les pods s'exécutent sur plusieurs nœuds pour s'assurer que l'application est toujours disponible, même si un travailleur tombe en panne.

- Nécessité de s'exécuter sur chaque nœud. Certains types d'applications Kubernetes sont destinés à s'exécuter dans le cluster lui-même sur chaque nœud maître ou travailleur. Les applications DNS et de surveillance sont des exemples d'applications qui doivent s'exécuter en continu sur chaque nœud. Vous pouvez exécuter ce type d'application en tant qu'ensemble de démons. Vous pouvez également exécuter un ensemble de démons sur un sous-ensemble de nœuds, en fonction des étiquettes de nœuds.

- Exiger une gestion du cycle de vie. Lorsque vous souhaitez céder votre application à d'autres personnes, envisagez de créer un opérateur. Les opérateurs vous permettent d'intégrer de l'intelligence, de sorte qu'ils peuvent gérer automatiquement des éléments tels que les sauvegardes et les mises à niveau. Associés au gestionnaire de cycle de vie des opérateurs (OLM), les gestionnaires de grappes peuvent exposer les opérateurs à des espaces de noms sélectionnés afin que les utilisateurs de la grappe puissent les exécuter.

- Avoir des exigences en matière d'identité ou de numérotation. Une application peut avoir des exigences en matière d'identité ou de numérotation. Par exemple, il peut vous être demandé d'exécuter exactement trois instances de l'application et de les nommer **0**, **1** et **2**. Un ensemble avec état convient à cette application. Les ensembles avec état sont particulièrement utiles pour les applications qui nécessitent un stockage indépendant, telles que les bases de données et les clusters zookeeper.

7.3.3. Composants de soutien disponibles

L'application que vous écrivez peut avoir besoin de composants de support, comme une base de données ou un composant de journalisation. Pour répondre à ce besoin, vous pourriez être en mesure d'obtenir le composant requis à partir des catalogues suivants qui sont disponibles dans la console web d'OpenShift Container Platform :

- OperatorHub, qui est disponible dans chaque cluster OpenShift Container Platform 4.12. L'OperatorHub met à la disposition de l'opérateur de cluster des opérateurs de Red Hat, des

partenaires certifiés de Red Hat et des membres de la communauté. L'opérateur de cluster peut rendre ces opérateurs disponibles dans tous les espaces de noms du cluster ou dans certains d'entre eux, afin que les développeurs puissent les lancer et les configurer avec leurs applications.

- Les modèles, qui sont utiles pour un type d'application unique, où le cycle de vie d'un composant n'est pas important après son installation. Un modèle fournit un moyen facile de commencer à développer une application Kubernetes avec un minimum de frais généraux. Un modèle peut être une liste de définitions de ressources, qui peuvent être **Deployment**, **Service**, **Route**, ou d'autres objets. Si vous souhaitez modifier les noms ou les ressources, vous pouvez définir ces valeurs en tant que paramètres dans le modèle.

Vous pouvez configurer les opérateurs et les modèles de support en fonction des besoins spécifiques de votre équipe de développement, puis les rendre disponibles dans les espaces de noms dans lesquels vos développeurs travaillent. De nombreuses personnes ajoutent des modèles partagés à l'espace de noms **openshift** car il est accessible depuis tous les autres espaces de noms.

7.3.4. Application du manifeste

Les manifestes Kubernetes vous permettent de créer une image plus complète des composants qui constituent vos applications Kubernetes. Vous écrivez ces manifestes sous forme de fichiers YAML et vous les déployez en les appliquant au cluster, par exemple en exécutant la commande **oc apply**.

7.3.5. Prochaines étapes

À ce stade, envisagez des moyens d'automatiser votre processus de développement de conteneurs. Idéalement, vous avez une sorte de pipeline CI qui construit les images et les pousse vers un registre. En particulier, un pipeline GitOps intègre votre développement de conteneurs avec les dépôts Git que vous utilisez pour stocker les logiciels nécessaires à la création de vos applications.

Le flux de travail jusqu'à ce point peut ressembler à ce qui suit :

- Jour 1 : Vous écrivez du YAML. Vous exécutez ensuite la commande **oc apply** pour appliquer ce YAML au cluster et tester son fonctionnement.
- Jour 2 : Vous mettez votre fichier de configuration de conteneur YAML dans votre propre dépôt Git. À partir de là, les personnes qui souhaitent installer cette application ou vous aider à l'améliorer peuvent extraire le fichier YAML et l'appliquer à leur cluster pour faire fonctionner l'application.
- Jour 3 : Envisagez de rédiger un opérateur pour votre candidature.

7.4. DÉVELOPPER POUR LES OPÉRATEURS

L'empaquetage et le déploiement de votre application en tant qu'opérateur peuvent être préférés si vous voulez que votre application soit disponible pour d'autres personnes. Comme indiqué précédemment, les opérateurs ajoutent un composant de cycle de vie à votre application qui reconnaît que le travail d'exécution d'une application n'est pas terminé dès qu'elle est installée.

Lorsque vous créez une application en tant qu'opérateur, vous pouvez intégrer vos propres connaissances en matière d'exécution et de maintenance de l'application. Vous pouvez intégrer des fonctionnalités permettant de mettre à jour l'application, de la sauvegarder, de la faire évoluer ou de garder une trace de son état. Si vous configurez correctement l'application, les tâches de maintenance, comme la mise à jour de l'opérateur, peuvent être effectuées automatiquement et de manière invisible pour les utilisateurs de l'opérateur.

Un exemple d'opérateur utile est celui qui est configuré pour sauvegarder automatiquement les données à des moments précis. Le fait qu'un opérateur gère la sauvegarde d'une application à des heures précises peut éviter à l'administrateur système de devoir s'en souvenir.

Toute maintenance d'application traditionnellement effectuée manuellement, comme la sauvegarde des données ou la rotation des certificats, peut être effectuée automatiquement par un opérateur.

CHAPITRE 8. RED HAT ENTERPRISE LINUX COREOS (RHCOS)

8.1. À PROPOS DU RHCOS

Red Hat Enterprise Linux CoreOS (RHCOS) représente la nouvelle génération de la technologie de système d'exploitation de conteneur à usage unique en offrant les normes de qualité de Red Hat Enterprise Linux (RHEL) avec des fonctions de mise à niveau automatisées et à distance.

RHCOS est pris en charge uniquement en tant que composant d'OpenShift Container Platform 4.12 pour toutes les machines OpenShift Container Platform. RHCOS est le seul système d'exploitation pris en charge pour les machines du plan de contrôle d'OpenShift Container Platform, ou machines maîtresses. Bien que RHCOS soit le système d'exploitation par défaut de toutes les machines de cluster, vous pouvez créer des machines de calcul, également appelées machines de travail, qui utilisent RHEL comme système d'exploitation. Il existe deux façons générales de déployer RHCOS dans OpenShift Container Platform 4.12 :

- Si vous installez votre cluster sur une infrastructure prévue par le programme d'installation, les images RHCOS sont téléchargées sur la plateforme cible pendant l'installation. Les fichiers de configuration Ignition appropriés, qui contrôlent la configuration RHCOS, sont également téléchargés et utilisés pour déployer les machines.

- Si vous installez votre cluster sur une infrastructure que vous gérez, vous devez suivre la documentation d'installation pour obtenir les images RHCOS, générer les fichiers de configuration Ignition et utiliser les fichiers de configuration Ignition pour provisionner vos machines.

8.1.1. Principales caractéristiques du RHCOS

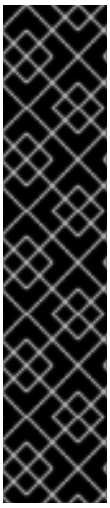
La liste suivante décrit les principales caractéristiques du système d'exploitation RHCOS :

- **Based on RHEL**: Le système d'exploitation sous-jacent est principalement constitué de composants RHEL. Les mesures de qualité, de sécurité et de contrôle qui soutiennent RHEL soutiennent également RHCOS. Par exemple, le logiciel RHCOS se présente sous la forme de paquets RPM, et chaque système RHCOS démarre avec un noyau RHEL et un ensemble de services gérés par le système init systemd.

- **Controlled immutability**: Bien qu'il contienne des composants RHEL, RHCOS est conçu pour être géré de manière plus étroite qu'une installation RHEL par défaut. La gestion est effectuée à distance depuis le cluster OpenShift Container Platform. Lorsque vous configurez vos machines RHCOS, vous ne pouvez modifier que quelques paramètres système. Cette immuabilité contrôlée permet à OpenShift Container Platform de stocker le dernier état des systèmes RHCOS dans le cluster afin d'être toujours en mesure de créer des machines supplémentaires et d'effectuer des mises à jour basées sur les dernières configurations RHCOS.

- **CRI-O container runtime**: Bien que RHCOS contienne des fonctionnalités permettant d'exécuter les conteneurs formatés OCI et libcontainer dont Docker a besoin, il intègre le moteur de conteneurs CRI-O au lieu du moteur de conteneurs Docker. En se concentrant sur les fonctionnalités nécessaires aux plateformes Kubernetes, telles que OpenShift Container Platform, CRI-O peut offrir une compatibilité spécifique avec les différentes versions de Kubernetes. CRI-O offre également une empreinte plus petite et une surface d'attaque réduite par rapport aux moteurs de conteneurs qui offrent un ensemble plus large de fonctionnalités. Pour l'instant, CRI-O est le seul moteur disponible dans les clusters d'OpenShift Container Platform.

CRI-O peut utiliser le runtime runC ou crun pour démarrer et gérer les conteneurs. Pour plus d'informations sur l'activation de crun, voir la documentation relative à la création d'un CR **ContainerRuntimeConfig**.



IMPORTANT

La prise en charge du moteur d'exécution de conteneur crun est une fonctionnalité d'aperçu technologique uniquement. Les fonctionnalités de l'aperçu technologique ne sont pas prises en charge par les accords de niveau de service (SLA) de production de Red Hat et peuvent ne pas être complètes sur le plan fonctionnel. Red Hat ne recommande pas leur utilisation en production. Ces fonctionnalités offrent un accès anticipé aux fonctionnalités des produits à venir, ce qui permet aux clients de tester les fonctionnalités et de fournir un retour d'information pendant le processus de développement.

Pour plus d'informations sur la portée de l'assistance des fonctionnalités de l'aperçu technologique de Red Hat, voir Portée de l'assistance des fonctionnalités de l'aperçu technologique.

- **Set of container tools**: Pour les tâches telles que la construction, la copie et la gestion des conteneurs, RHCOS remplace l'outil Docker CLI par un ensemble compatible d'outils de conteneurs. L'outil CLI podman prend en charge de nombreuses fonctionnalités d'exécution de conteneurs, telles que l'exécution, le démarrage, l'arrêt, l'énumération et la suppression de conteneurs et d'images de conteneurs. L'outil CLI skopeo permet de copier, d'authentifier et de signer des images. Vous pouvez utiliser l'outil CLI **crictl** pour travailler avec les conteneurs et les pods du moteur de conteneurs CRI-O. Bien que l'utilisation directe de ces outils dans RHCOS soit déconseillée, vous pouvez les utiliser à des fins de débogage.

- **rpm-ostree upgrades**: RHCOS propose des mises à jour transactionnelles à l'aide du système **rpm-ostree**. Les mises à jour sont fournies au moyen d'images de conteneurs et font partie du processus de mise à jour d'OpenShift Container Platform. Lors du déploiement, l'image du conteneur est extraite et écrite sur le disque, puis le chargeur de démarrage est modifié pour démarrer avec la nouvelle version. La machine redémarre dans la mise à jour de manière continue pour s'assurer que la capacité du cluster est minimalement impactée.

- **bootupd firmware and bootloader updater**: Les gestionnaires de paquets et les systèmes hybrides tels que **rpm-ostree** ne mettent pas à jour le micrologiciel ou le chargeur de démarrage. Avec **bootupd**, les utilisateurs de RHCOS ont accès à un outil de mise à jour indépendant de toute distribution et de tout système qui gère les mises à jour du micrologiciel et du chargeur de démarrage dans les modes de démarrage UEFI et BIOS hérités qui fonctionnent sur des architectures modernes, telles que x86_64, ppc64le et aarch64.
  Pour plus d'informations sur l'installation de **bootupd**, voir la documentation de *Updating the bootloader using bootupd*.

- **Updated through the Machine Config Operator**: Dans OpenShift Container Platform, le Machine Config Operator gère les mises à niveau du système d'exploitation. Au lieu de mettre à niveau des paquets individuels, comme c'est le cas avec les mises à niveau **yum**, **rpm-ostree** fournit des mises à niveau du système d'exploitation en tant qu'unité atomique. Le déploiement du nouveau système d'exploitation est mis en scène pendant les mises à niveau et entre en vigueur au prochain redémarrage. Si un problème survient lors de la mise à niveau, un simple retour en arrière et un redémarrage ramènent le système à l'état précédent. Les mises à niveau du RHCOS dans OpenShift Container Platform sont effectuées lors des mises à jour des clusters.

Pour les systèmes RHCOS, la disposition du système de fichiers **rpm-ostree** présente les caractéristiques suivantes :

- /**usr** est l'endroit où sont stockés les binaires et les bibliothèques du système d'exploitation et est en lecture seule. Nous ne sommes pas favorables à sa modification.

- /**etc**, /**boot**, /**var** sont accessibles en écriture sur le système mais ne doivent être modifiés que par l'opérateur de la configuration de la machine.

- /**var**/**lib**/**containers** est l'emplacement de stockage du graphique pour les images des conteneurs.

8.1.2. Choix de la configuration de RHCOS

RHCOS est conçu pour se déployer sur un cluster OpenShift Container Platform avec un minimum de configuration de la part de l'utilisateur. Dans sa forme la plus basique, cela consiste à :

- Commencer par une infrastructure provisionnée, comme sur AWS, ou provisionner soi-même l'infrastructure.

- Fournir quelques informations, telles que les informations d'identification et le nom du cluster, dans un fichier **install-config.yaml** lors de l'exécution de **openshift-install**.

Parce que les systèmes RHCOS dans OpenShift Container Platform sont conçus pour être entièrement gérés à partir du cluster OpenShift Container Platform par la suite, il est déconseillé de modifier directement une machine RHCOS. Bien qu'un accès direct limité au cluster de machines RHCOS puisse être réalisé à des fins de débogage, vous ne devriez pas configurer directement les systèmes RHCOS. Au lieu de cela, si vous avez besoin d'ajouter ou de modifier des fonctionnalités sur vos nœuds OpenShift Container Platform, envisagez d'apporter des modifications de la manière suivante :

- **Kubernetes workload objects, such as DaemonSet and Deployment** Si vous devez ajouter des services ou d'autres fonctionnalités de niveau utilisateur à votre cluster, envisagez de les ajouter en tant qu'objets de charge de travail Kubernetes. Garder ces fonctionnalités en dehors des configurations de nœuds spécifiques est le meilleur moyen de réduire le risque de casser le cluster lors des mises à niveau ultérieures.

- **Day-2 customizations**: Dans la mesure du possible, mettez en place une grappe sans personnaliser les nœuds de la grappe et apportez les modifications nécessaires aux nœuds après la mise en place de la grappe. Ces modifications sont plus faciles à suivre ultérieurement et risquent moins d'interrompre les mises à jour. La création de configurations de machines ou la modification des ressources personnalisées de l'opérateur sont des moyens d'effectuer ces personnalisations.

- **Day-1 customizations**: Pour les personnalisations que vous devez mettre en œuvre lors du démarrage du cluster, il existe des moyens de modifier votre cluster afin que les changements soient mis en œuvre lors du premier démarrage. Les personnalisations du jour 1 peuvent être effectuées par le biais des configurations Ignition et des fichiers manifestes pendant **openshift-install** ou en ajoutant des options de démarrage pendant les installations ISO fournies par l'utilisateur.

Voici des exemples de personnalisations que vous pouvez effectuer dès le premier jour :

- **Kernel arguments**: Si des fonctionnalités ou des réglages particuliers du noyau sont nécessaires sur les nœuds lors du premier démarrage de la grappe.

- **Disk encryption**: Si vos besoins en matière de sécurité exigent que le système de fichiers racine des nœuds soit crypté, par exemple avec la prise en charge FIPS.

- **Kernel modules**: Si un périphérique particulier, tel qu'une carte réseau ou une carte vidéo, ne dispose pas d'un module utilisable par défaut dans le noyau Linux.

- **Chronyd**: Si vous souhaitez fournir des paramètres d'horloge spécifiques à vos nœuds, tels que l'emplacement des serveurs de temps.

Pour accomplir ces tâches, vous pouvez augmenter le processus **openshift-install** pour inclure des objets supplémentaires tels que les objets **MachineConfig**. Les procédures qui aboutissent à la création de configurations de machines peuvent être transmises à l'opérateur de configuration de machines une fois que la grappe est en place.

NOTE

- Les fichiers de configuration d'Ignition générés par le programme d'installation contiennent des certificats qui expirent après 24 heures et qui sont renouvelés à ce moment-là. Si le cluster est arrêté avant le renouvellement des certificats et qu'il est redémarré après l'expiration des 24 heures, le cluster récupère automatiquement les certificats expirés. L'exception est que vous devez approuver manuellement les demandes de signature de certificat (CSR) de **node-bootstrapper** en attente pour récupérer les certificats de kubelet. Pour plus d'informations, consultez la documentation relative à *Recovering from expired control plane certificates*.

- Il est recommandé d'utiliser les fichiers de configuration Ignition dans les 12 heures suivant leur génération, car le certificat de 24 heures tourne entre 16 et 22 heures après l'installation du cluster. En utilisant les fichiers de configuration Ignition dans les 12 heures, vous pouvez éviter l'échec de l'installation si la mise à jour du certificat s'exécute pendant l'installation.

8.1.3. Choisir comment déployer RHCOS

Les différences entre les installations RHCOS pour OpenShift Container Platform sont basées sur le fait que vous déployez sur une infrastructure provisionnée par le programme d'installation ou par l'utilisateur :

- **Installer-provisioned**: Certains environnements cloud offrent des infrastructures préconfigurées qui vous permettent de mettre en place un cluster OpenShift Container Platform avec une configuration minimale. Pour ces types d'installations, vous pouvez fournir des configurations Ignition qui placent le contenu sur chaque nœud afin qu'il soit présent lorsque le cluster démarre pour la première fois.

- **User-provisioned**: Si vous provisionnez votre propre infrastructure, vous disposez d'une plus grande flexibilité dans la manière d'ajouter du contenu à un nœud RHCOS. Par exemple, vous pouvez ajouter des arguments de noyau lorsque vous démarrez le programme d'installation ISO RHCOS pour installer chaque système. Cependant, dans la plupart des cas où la configuration est requise sur le système d'exploitation lui-même, il est préférable de fournir cette configuration par le biais d'une configuration Ignition.

La fonction Ignition ne fonctionne que lorsque le système RHCOS est configuré pour la première fois. Ensuite, les configurations d'allumage peuvent être fournies ultérieurement en utilisant la configuration de la machine.

8.1.4. À propos d'Ignition

Ignition est l'utilitaire utilisé par RHCOS pour manipuler les disques lors de la configuration initiale. Il effectue des tâches courantes sur les disques, notamment le partitionnement des disques, le formatage des partitions, l'écriture de fichiers et la configuration des utilisateurs. Au premier démarrage, Ignition lit

sa configuration à partir du média d'installation ou de l'emplacement que vous avez spécifié et applique la configuration aux machines.

Que vous installiez votre cluster ou que vous y ajoutiez des machines, Ignition effectue toujours la configuration initiale des machines du cluster OpenShift Container Platform. La majeure partie de la configuration réelle du système se produit sur chaque machine elle-même. Pour chaque machine, Ignition prend l'image RHCOS et démarre le noyau RHCOS. Les options de la ligne de commande du noyau identifient le type de déploiement et l'emplacement du disque RAM initial activé par Ignition (initramfs).

8.1.4.1. Comment fonctionne l'allumage

Pour créer des machines en utilisant Ignition, vous avez besoin de fichiers de configuration Ignition. Le programme d'installation d'OpenShift Container Platform crée les fichiers de configuration Ignition dont vous avez besoin pour déployer votre cluster. Ces fichiers sont basés sur les informations que vous fournissez au programme d'installation directement ou par le biais d'un fichier **install-config.yaml**.

La façon dont Ignition configure les machines est similaire à la façon dont des outils comme cloud-init ou Linux Anaconda kickstart configurent les systèmes, mais avec quelques différences importantes :

- Ignition fonctionne à partir d'un disque RAM initial qui est séparé du système sur lequel vous effectuez l'installation. Ignition peut donc repartitionner les disques, configurer les systèmes de fichiers et apporter d'autres modifications au système de fichiers permanent de la machine. En revanche, cloud-init s'exécute dans le cadre d'un système d'initialisation de la machine lorsque le système démarre, de sorte qu'il n'est pas aussi facile d'apporter des modifications fondamentales à des éléments tels que les partitions de disque. Avec cloud-init, il est également difficile de reconfigurer le processus de démarrage lorsque vous êtes au milieu du processus de démarrage du nœud.

- Ignition est destiné à initialiser les systèmes, et non à modifier les systèmes existants. Après l'initialisation d'une machine et l'exécution du noyau à partir du système installé, l'opérateur de configuration de machine du cluster OpenShift Container Platform complète toute la configuration future de la machine.

- Au lieu de réaliser un ensemble défini d'actions, Ignition met en œuvre une configuration déclarative. Il vérifie que toutes les partitions, fichiers, services et autres éléments sont en place avant le démarrage de la nouvelle machine. Il effectue ensuite les modifications, comme la copie de fichiers sur le disque, qui sont nécessaires pour que la nouvelle machine réponde à la configuration spécifiée.

- Une fois qu'Ignition a fini de configurer une machine, le noyau continue de tourner mais abandonne le disque RAM initial et pivote vers le système installé sur le disque. Tous les nouveaux services et autres fonctionnalités du système démarrent sans qu'il soit nécessaire de redémarrer le système.

- Comme Ignition confirme que toutes les nouvelles machines répondent à la configuration déclarée, vous ne pouvez pas avoir une machine partiellement configurée. Si la configuration d'une machine échoue, le processus d'initialisation ne se termine pas et Ignition ne démarre pas la nouvelle machine. Votre cluster ne contiendra jamais de machines partiellement configurées. Si Ignition ne peut pas se terminer, la machine n'est pas ajoutée au cluster. Vous devez ajouter une nouvelle machine à la place. Ce comportement permet d'éviter le cas difficile du débogage d'une machine lorsque les résultats d'une tâche de configuration échouée ne sont pas connus jusqu'à ce que quelque chose qui en dépende tombe en panne à une date ultérieure.

- Si un problème avec une configuration Ignition fait échouer la configuration d'une machine, Ignition n'essaiera pas d'utiliser la même configuration pour configurer une autre machine. Par

exemple, un échec pourrait résulter d'une configuration Ignition composée d'une configuration parent et d'une configuration enfant qui veulent toutes deux créer le même fichier. Dans ce cas, un échec empêcherait d'utiliser à nouveau cette configuration Ignition pour configurer une autre machine jusqu'à ce que le problème soit résolu.

- Si vous avez plusieurs fichiers de configuration Ignition, vous obtenez une union de cet ensemble de configurations. Ignition étant déclaratif, des conflits entre les fichiers de configuration pourraient empêcher Ignition de configurer la machine. L'ordre des informations dans ces fichiers n'a pas d'importance. Ignition triera et implémentera chaque paramètre de la manière la plus logique possible. Par exemple, si un fichier a besoin d'un répertoire à plusieurs niveaux de profondeur et qu'un autre fichier a besoin d'un répertoire le long de ce chemin, le dernier fichier est créé en premier. Ignition trie et crée tous les fichiers, répertoires et liens par profondeur.

- Parce qu'Ignition peut démarrer avec un disque dur complètement vide, il peut faire quelque chose que cloud-init ne peut pas faire : configurer des systèmes sur du métal nu à partir de zéro en utilisant des fonctionnalités telles que le démarrage PXE. Dans le cas du bare metal, la configuration Ignition est injectée dans la partition de démarrage afin qu'Ignition puisse la trouver et configurer le système correctement.

8.1.4.2. La séquence d'allumage

Le processus d'ignition d'une machine RHCOS dans un cluster OpenShift Container Platform comprend les étapes suivantes :

- La machine obtient son fichier de configuration Ignition. Les machines du plan de contrôle obtiennent leurs fichiers de configuration Ignition à partir de la machine de démarrage, et les machines de travail obtiennent les fichiers de configuration Ignition à partir d'une machine du plan de contrôle.

- Ignition crée des partitions de disque, des systèmes de fichiers, des répertoires et des liens sur la machine. Il prend en charge les matrices RAID, mais pas les volumes LVM.

- Ignition monte la racine du système de fichiers permanent dans le répertoire **/sysroot** de l'initramfs et commence à travailler dans ce répertoire **/sysroot**.

- Ignition configure tous les systèmes de fichiers définis et les configure pour qu'ils soient montés de manière appropriée au moment de l'exécution.

- Ignition exécute les fichiers temporaires **systemd** pour remplir les fichiers requis dans le répertoire **/var**.

- Ignition exécute les fichiers de configuration Ignition pour configurer les utilisateurs, les fichiers unitaires systemd et d'autres fichiers de configuration.

- L'allumage démonte tous les composants du système permanent qui ont été montés dans l'initramfs.

- Ignition démarre le processus init de la nouvelle machine, qui démarre à son tour tous les autres services de la machine qui s'exécutent pendant le démarrage du système.

À la fin de ce processus, la machine est prête à rejoindre le cluster et ne nécessite pas de redémarrage.

8.2. VISUALISATION DES FICHIERS DE CONFIGURATION D'IGNITION

Pour voir le fichier de configuration Ignition utilisé pour déployer la machine d'amorçage, exécutez la commande suivante :

```
$ openshift-install create ignition-configs --dir $HOME/testconfig
```

Après avoir répondu à quelques questions, les fichiers **bootstrap.ign**, **master.ign** et **worker.ign** apparaissent dans le répertoire que vous avez indiqué.

Pour voir le contenu du fichier **bootstrap.ign**, faites-le passer par le filtre **jq**. Voici un extrait de ce fichier :

```
$ cat $HOME/testconfig/bootstrap.ign | jq
{
  "ignition": {
    "version": "3.2.0"
  },
  "passwd": {
    "users": [
      {
        "name": "core",
        "sshAuthorizedKeys": [
          "ssh-rsa AAAAB3NzaC1yc...."
        ]
      }
    ]
  },
  "storage": {
    "files": [
      {
        "overwrite": false,
        "path": "/etc/motd",
        "user": {
          "name": "root"
        },
        "append": [
          {
            "source": "data:text/plain;charset=utf-
8;base64,VGhpcyBpcyB0aGUgYm9vdHN0cmFwIG5vZGU7IGl0IHdpbGwgYmUgZGVzdHJveWVkIHdo
ZW4gdGhlIG1hc3RlciBpcyBmdWxseSB1cC4KClRoZSBwcmltYXJ5IHNlcnZpY2VzIGFyZSByZWxlYXNlI
WltYWdlLnNlcnZpY2UgZm9sbG93ZWQgYnkgYm9vdGt1YmUuc2VydmljZS4gVG8gd2F0Y2ggdGhlaXI
gc3RhdHVzLCBydW4gZS5nLgoKICBqb3VybmFsY3RsIC1iIC1mIC11IHJlbGVhc2UtaW1hZ2Uuc2Vydm
ljZSAtdSBib290a3ViZS5zZXJ2aWNlCg=="
          }
        ],
        "mode": 420
      },
...
```

Pour décoder le contenu d'un fichier répertorié dans le fichier **bootstrap.ign**, envoyez la chaîne de données codée en base64 représentant le contenu de ce fichier à la commande **base64 -d**. Voici un exemple utilisant le contenu du fichier **/etc/motd** ajouté à la machine d'amorçage à partir de la sortie indiquée ci-dessus :

```
$ echo
VGhpcyBpcyB0aGUgYm9vdHN0cmFwIG5vZGU7IGl0IHdpbGwgYmUgZGVzdHJveWVkIHdoZW4gdG
```

```
hIIG1hc3RlciBpcyBmdWxseSB1cC4KClRoZSBwcmltYXJ5IHNlcnZpY2VzIGFyZSByZWxlYXNlLWltYWdl
LnNlcnZpY2UgZm9sbG93ZWQgYnkgYm9vdGt1YmUuc2VydmljZS4gVG8gd2F0Y2ggdGhlaXIgc3Rhd
HVzLCBydW4gZS5nLgoKICBqb3VybmFsY3RsIC1iIC1mIC11IHJlbGVhc2UtaW1hZ2Uuc2VydmljZSAtd
SBib290a3ViZS5zZXJ2aWNlCg== | base64 --decode
```

Exemple de sortie

```
This is the bootstrap node; it will be destroyed when the master is fully up.

The primary services are release-image.service followed by bootkube.service. To watch their status, run e.g.

  journalctl -b -f -u release-image.service -u bootkube.service
```

Répétez ces commandes sur les fichiers **master.ign** et **worker.ign** pour voir la source des fichiers de configuration Ignition pour chacun de ces types de machines. Vous devriez voir une ligne comme la suivante pour le fichier **worker.ign**, identifiant comment il obtient sa configuration Ignition à partir de la machine bootstrap :

```
"source": "https://api.myign.develcluster.example.com:22623/config/worker",
```

Voici quelques enseignements à tirer du dossier **bootstrap.ign**:

- Format : Le format du fichier est défini dans la spécification de configuration d'Ignition. Les fichiers de même format sont utilisés ultérieurement par le MCO pour fusionner les changements dans la configuration d'une machine.
- Contenu : Comme la machine d'amorçage sert les configurations Ignition des autres machines, les informations de configuration Ignition des machines maître et ouvrière sont stockées dans le site **bootstrap.ign**, avec la configuration de la machine d'amorçage.
- Taille : Le fichier comporte plus de 1300 lignes, avec des chemins d'accès à différents types de ressources.
- Le contenu de chaque fichier qui sera copié sur la machine est en fait encodé dans des URL de données, ce qui tend à rendre le contenu un peu difficile à lire. (Utilisez les commandes **jq** et **base64** présentées précédemment pour rendre le contenu plus lisible)
- Configuration : Les différentes sections du fichier de configuration d'Ignition sont généralement destinées à contenir des fichiers qui sont simplement déposés dans le système de fichiers d'une machine, plutôt que des commandes pour modifier des fichiers existants. Par exemple, au lieu d'avoir une section sur NFS qui configure ce service, vous pouvez simplement ajouter un fichier de configuration NFS, qui sera ensuite lancé par le processus init au démarrage du système.
- utilisateurs : Un utilisateur nommé **core** est créé, avec votre clé SSH assignée à cet utilisateur. Cela vous permet de vous connecter au cluster avec ce nom d'utilisateur et vos informations d'identification.
- le stockage : La section stockage identifie les fichiers qui sont ajoutés à chaque machine. Parmi les fichiers les plus importants, citons **/root/.docker/config.json** (qui fournit les informations d'identification dont votre cluster a besoin pour puiser dans les registres d'images de conteneurs) et un certain nombre de fichiers manifestes dans **/opt/openshift/manifests** qui sont utilisés pour configurer votre cluster.

- systemd : La section **systemd** contient le contenu utilisé pour créer les fichiers unitaires **systemd**. Ces fichiers sont utilisés pour démarrer les services au moment du démarrage, ainsi que pour gérer ces services sur les systèmes en cours d'exécution.
- Primitives : Ignition expose également des primitives de bas niveau sur lesquelles d'autres outils peuvent s'appuyer.

8.3. MODIFICATION DE LA CONFIGURATION DE L'ALLUMAGE APRÈS L'INSTALLATION

Les pools de configuration de machines gèrent une grappe de nœuds et leurs configurations de machines correspondantes. Les configurations de machines contiennent des informations sur la configuration d'une grappe. Pour dresser la liste de tous les pools de configuration de machines connus :

```
$ oc get machineconfigpools
```

Exemple de sortie

```
NAME   CONFIG                                 UPDATED UPDATING DEGRADED
master master-1638c1aea398413bb918e76632f20799 False   False    False
worker worker-2feef4f8288936489a5a832ca8efe953 False   False    False
```

Pour lister toutes les configurations des machines :

```
$ oc get machineconfig
```

Exemple de sortie

```
NAME                                       GENERATEDBYCONTROLLER   IGNITIONVERSION   CREATED
OSIMAGEURL

00-master                                  4.0.0-0.150.0.0-dirty   3.2.0             16m
00-master-ssh                              4.0.0-0.150.0.0-dirty                     16m
00-worker                                  4.0.0-0.150.0.0-dirty   3.2.0             16m
00-worker-ssh                              4.0.0-0.150.0.0-dirty                     16m
01-master-kubelet                          4.0.0-0.150.0.0-dirty   3.2.0             16m
01-worker-kubelet                          4.0.0-0.150.0.0-dirty   3.2.0             16m
master-1638c1aea398413bb918e76632f20799    4.0.0-0.150.0.0-dirty   3.2.0             16m
worker-2feef4f8288936489a5a832ca8efe953    4.0.0-0.150.0.0-dirty   3.2.0             16m
```

L'opérateur de configuration de la machine agit quelque peu différemment de l'allumage lorsqu'il s'agit d'appliquer ces configurations de la machine. Les configurations de machine sont lues dans l'ordre (de 00* à 99*). Les étiquettes à l'intérieur des configurations de machine identifient le type de nœud auquel elles sont destinées (maître ou travailleur). Si le même fichier apparaît dans plusieurs fichiers de configuration de machine, c'est le dernier qui l'emporte. Ainsi, par exemple, tout fichier apparaissant dans un fichier 99* remplacera le même fichier apparaissant dans un fichier 00*. Les objets **MachineConfig** en entrée sont réunis en un objet **MachineConfig** "rendu", qui sera utilisé comme cible par l'opérateur et qui est la valeur que vous pouvez voir dans le pool de configuration de la machine.

Pour savoir quels fichiers sont gérés à partir d'une configuration de machine, recherchez "Path:" à l'intérieur d'un objet **MachineConfig** particulier. Par exemple :

```
$ oc describe machineconfigs 01-worker-container-runtime | grep Path:
```

Exemple de sortie

```
Path:           /etc/containers/registries.conf
Path:           /etc/containers/storage.conf
Path:           /etc/crio/crio.conf
```

Veillez à donner au fichier de configuration de la machine un nom ultérieur (tel que 10-worker-container-runtime). Gardez à l'esprit que le contenu de chaque fichier se présente sous la forme de données de type URL. Appliquez ensuite la nouvelle configuration machine au cluster.

CHAPITRE 9. PLUGINS D'ADMISSION

9.1. A PROPOS DES PLUGINS D'ADMISSION

Les plugins d'admission sont utilisés pour aider à réguler le fonctionnement d'OpenShift Container Platform 4.12. Les plugins d'admission interceptent les demandes adressées à l'API principale pour valider les demandes de ressources et s'assurer que les politiques sont respectées, après que la demande a été authentifiée et autorisée. Par exemple, ils sont couramment utilisés pour appliquer la politique de sécurité, les limitations de ressources ou les exigences de configuration.

Les plugins d'admission s'exécutent en séquence comme une chaîne d'admission. Si l'un des modules d'admission de la séquence rejette une demande, la chaîne entière est interrompue et une erreur est renvoyée.

OpenShift Container Platform dispose d'un ensemble de plugins d'admission activés par défaut pour chaque type de ressource. Ces plugins sont nécessaires au bon fonctionnement du cluster. Les plugins d'admission ignorent les ressources dont ils ne sont pas responsables.

Outre les valeurs par défaut, la chaîne d'admission peut être étendue de manière dynamique grâce à des plugins d'admission de type webhook qui font appel à des serveurs webhook personnalisés. Il existe deux types de plugins d'admission aux webhooks : un plugin d'admission à la mutation et un plugin d'admission à la validation. Le plugin d'admission à la mutation s'exécute en premier et peut à la fois modifier les ressources et valider les demandes. Le plugin de validation des admissions valide les demandes et s'exécute après le plugin de validation des admissions afin que les modifications déclenchées par le plugin de validation des admissions puissent également être validées.

L'appel à des serveurs de webhook par l'intermédiaire d'un plugin d'admission mutant peut produire des effets secondaires sur les ressources liées à l'objet cible. Dans de telles situations, vous devez prendre des mesures pour valider que le résultat final est conforme aux attentes.

AVERTISSEMENT

L'admission dynamique doit être utilisée avec prudence car elle a un impact sur les opérations du plan de contrôle du cluster. Lorsque vous appelez des serveurs webhook via des plugins d'admission webhook dans OpenShift Container Platform 4.12, assurez-vous d'avoir lu entièrement la documentation et d'avoir testé les effets secondaires des mutations. Inclure des étapes pour restaurer les ressources à leur état d'origine avant la mutation, dans le cas où une demande ne passe pas par l'ensemble de la chaîne d'admission.

9.2. PLUGINS D'ADMISSION PAR DÉFAUT

Les plugins de validation et d'admission par défaut sont activés dans OpenShift Container Platform 4.12. Ces plugins par défaut contribuent aux fonctionnalités fondamentales du plan de contrôle, telles que la politique d'entrée, le dépassement des limites de ressources du cluster et la politique de quotas. Les listes suivantes contiennent les plugins d'admission par défaut :

Exemple 9.1. Validation des plugins d'admission

- **LimitRanger**

- **ServiceAccount**
- **PodNodeSelector**
- **Priority**
- **PodTolerationRestriction**
- **OwnerReferencesPermissionEnforcement**
- **PersistentVolumeClaimResize**
- **RuntimeClass**
- **CertificateApproval**
- **CertificateSigning**
- **CertificateSubjectRestriction**
- **autoscaling.openshift.io/ManagementCPUsOverride**
- **authorization.openshift.io/RestrictSubjectBindings**
- **scheduling.openshift.io/OriginPodNodeEnvironment**
- **network.openshift.io/ExternalIPRanger**
- **network.openshift.io/RestrictedEndpointsAdmission**
- **image.openshift.io/ImagePolicy**
- **security.openshift.io/SecurityContextConstraint**
- **security.openshift.io/SCCExecRestrictions**
- **route.openshift.io/IngressAdmission**
- **config.openshift.io/ValidateAPIServer**
- **config.openshift.io/ValidateAuthentication**
- **config.openshift.io/ValidateFeatureGate**
- **config.openshift.io/ValidateConsole**
- **operator.openshift.io/ValidateDNS**
- **config.openshift.io/ValidateImage**
- **config.openshift.io/ValidateOAuth**
- **config.openshift.io/ValidateProject**
- **config.openshift.io/DenyDeleteClusterConfiguration**
- **config.openshift.io/ValidateScheduler**

- **quota.openshift.io/ValidateClusterResourceQuota**
- **security.openshift.io/ValidateSecurityContextConstraints**
- **authorization.openshift.io/ValidateRoleBindingRestriction**
- **config.openshift.io/ValidateNetwork**
- **operator.openshift.io/ValidateKubeControllerManager**
- **ValidatingAdmissionWebhook**
- **ResourceQuota**
- **quota.openshift.io/ClusterResourceQuota**

Exemple 9.2. Mutation des plugins d'admission

- **NamespaceLifecycle**
- **LimitRanger**
- **ServiceAccount**
- **NodeRestriction**
- **TaintNodesByCondition**
- **PodNodeSelector**
- **Priority**
- **DefaultTolerationSeconds**
- **PodTolerationRestriction**
- **DefaultStorageClass**
- **StorageObjectInUseProtection**
- **RuntimeClass**
- **DefaultIngressClass**
- **autoscaling.openshift.io/ManagementCPUsOverride**
- **scheduling.openshift.io/OriginPodNodeEnvironment**
- **image.openshift.io/ImagePolicy**
- **security.openshift.io/SecurityContextConstraint**
- **security.openshift.io/DefaultSecurityContextConstraints**
- **MutatingAdmissionWebhook**

9.3. PLUGINS D'ADMISSION AUX WEBHOOKS

En plus des plugins d'admission par défaut d'OpenShift Container Platform, l'admission dynamique peut être mise en œuvre via des plugins d'admission webhook qui appellent des serveurs webhook, afin d'étendre la fonctionnalité de la chaîne d'admission. Les serveurs webhook sont appelés via HTTP à des points d'extrémité définis.

Il existe deux types de plugins d'admission de webhook dans OpenShift Container Platform :

- Au cours du processus d'admission, le site *mutating admission plugin* peut effectuer des tâches telles que l'injection d'étiquettes d'affinité.
- À la fin du processus d'admission, le site *validating admission plugin* peut être utilisé pour s'assurer qu'un objet est correctement configuré, par exemple en s'assurant que les étiquettes d'affinité sont conformes aux attentes. Si la validation passe, OpenShift Container Platform planifie l'objet tel qu'il est configuré.

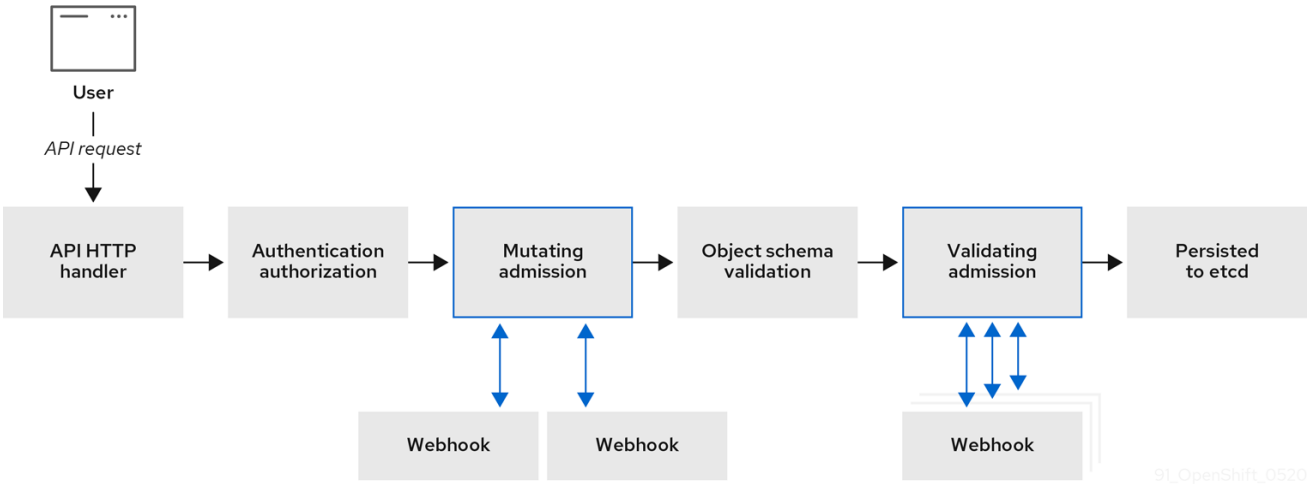
Lorsqu'une demande d'API arrive, les plugins d'admission de mutation ou de validation utilisent la liste des webhooks externes dans la configuration et les appellent en parallèle :

- Si tous les webhooks approuvent la demande, la chaîne d'admission se poursuit.
- Si l'un des webhooks refuse la demande, la demande d'admission est refusée et la raison de ce refus est basée sur le premier refus.
- Si plusieurs webhooks refusent la demande d'admission, seul le premier motif de refus est renvoyé à l'utilisateur.
- En cas d'erreur lors de l'appel d'un webhook, la demande est refusée ou le webhook est ignoré en fonction de la politique d'erreur définie. Si la politique d'erreur est définie sur **Ignore**, la demande est acceptée sans condition en cas d'échec. Si la politique est définie sur **Fail**, les demandes qui échouent sont refusées. L'utilisation de **Ignore** peut entraîner un comportement imprévisible pour tous les clients.

La communication entre le plugin webhook admission et le serveur webhook doit utiliser TLS. Générez un certificat d'autorité de certification et utilisez-le pour signer le certificat de serveur utilisé par votre serveur d'admission au webhook. Le certificat d'autorité de certification codé en PEM est fourni au plugin de webhook admission à l'aide d'un mécanisme tel que le service serving certificate secrets.

Le diagramme suivant illustre le processus séquentiel de la chaîne d'admission dans lequel plusieurs serveurs de webhook sont appelés.

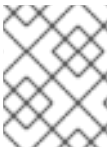
Figure 9.1. Chaîne d'admission de l'API avec des plugins d'admission mutants et validants



Un exemple de cas d'utilisation du plugin d'admission webhook est celui où tous les pods doivent avoir un ensemble commun d'étiquettes. Dans cet exemple, le plugin d'admission mutant peut injecter des étiquettes et le plugin d'admission validant peut vérifier que les étiquettes sont conformes aux attentes. OpenShift Container Platform planifierait ensuite les pods qui incluent les labels requis et rejetterait ceux qui ne les incluent pas.

Voici quelques cas d'utilisation courants des plugins d'admission aux webhooks :

- Réservation d'espace de noms.
- Limitation des ressources réseau personnalisées gérées par le plugin de périphérique réseau SR-IOV.
- Définition de tolérances permettant aux taints de déterminer quels pods doivent être planifiés sur un nœud.
- Validation de la classe de priorité des pods.



NOTE

La valeur par défaut du délai d'attente du webhook dans OpenShift Container Platform est de 13 secondes et ne peut pas être modifiée.

9.4. TYPES DE PLUGINS D'ADMISSION AUX WEBHOOKS

Les administrateurs de clusters peuvent faire appel à des serveurs webhook par l'intermédiaire du plugin d'admission de mutation ou du plugin d'admission de validation dans la chaîne d'admission du serveur API.

9.4.1. Mutation du plugin d'admission

Le plugin d'admission à la mutation est invoqué pendant la phase de mutation du processus d'admission, ce qui permet de modifier le contenu de la ressource avant qu'elle ne soit conservée. Un exemple de webhook qui peut être appelé par le plugin d'admission à la mutation est la fonctionnalité Pod Node Selector, qui utilise une annotation sur un espace de noms pour trouver un sélecteur d'étiquette et l'ajouter à la spécification du pod.

Exemple de configuration du plugin d'admission à la mutation

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: MutatingWebhookConfiguration (1)
metadata:
  name: <webhook_name> (2)
webhooks:
- name: <webhook_name> (3)
  clientConfig: (4)
    service:
      namespace: default (5)
      name: kubernetes (6)
      path: <webhook_url> (7)
    caBundle: <ca_signing_certificate> (8)
  rules: (9)
  - operations: (10)
    - <operation>
    apiGroups:
    - ""
    apiVersions:
    - "*"
    resources:
    - <resource>
  failurePolicy: <policy> (11)
  sideEffects: None
```

(1) Spécifie une configuration de plugin d'admission à la mutation.

(2) Le nom de l'objet **MutatingWebhookConfiguration**. Remplacer **<webhook_name>** par la valeur appropriée.

(3) Le nom du webhook à appeler. Remplacez **<webhook_name>** par la valeur appropriée.

(4) Informations sur la manière de se connecter au serveur webhook, de lui faire confiance et de lui envoyer des données.

(5) L'espace de noms dans lequel le service frontal est créé.

(6) Le nom du service frontal.

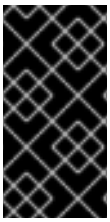
(7) URL du webhook utilisé pour les demandes d'admission. Remplacez **<webhook_url>** par la valeur appropriée.

(8) Un certificat d'autorité de certification codé en PEM qui signe le certificat de serveur utilisé par le serveur webhook. Remplacez **<ca_signing_certificate>** par le certificat approprié au format base64.

(9) Règles qui définissent quand le serveur API doit utiliser ce plugin d'admission au webhook.

(10) Une ou plusieurs opérations qui déclenchent l'appel du serveur API à ce plugin d'admission au webhook. Les valeurs possibles sont **create**, **update**, **delete** ou **connect**. Remplacez **<operation>** et **<resource>** par les valeurs appropriées.

(11) Indique comment la politique doit se dérouler si le serveur webhook n'est pas disponible. Remplacez **<policy>** par **Ignore** (pour accepter inconditionnellement la demande en cas d'échec) ou **Fail** (pour refuser la demande qui a échoué). L'utilisation de **Ignore** peut entraîner un comportement imprévisible pour tous les clients.



IMPORTANT

Dans OpenShift Container Platform 4.12, les objets créés par des utilisateurs ou des boucles de contrôle via un plugin d'admission mutant peuvent renvoyer des résultats inattendus, notamment si les valeurs définies dans une requête initiale sont écrasées, ce qui n'est pas recommandé.

9.4.2. Validation du plugin d'admission

Un plugin d'admission validant est invoqué pendant la phase de validation du processus d'admission. Cette phase permet d'appliquer des invariants à des ressources API particulières afin de s'assurer que la ressource ne sera pas modifiée à nouveau. Le Pod Node Selector est également un exemple de webhook appelé par le plugin de validation d'admission, pour s'assurer que tous les champs **nodeSelector** sont contraints par les restrictions du sélecteur de nœuds sur l'espace de noms.

Exemple de validation de la configuration du plugin d'admission

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingWebhookConfiguration (1)
metadata:
  name: <webhook_name> (2)
webhooks:
- name: <webhook_name> (3)
  clientConfig: (4)
    service:
      namespace: default (5)
      name: kubernetes (6)
      path: <webhook_url> (7)
    caBundle: <ca_signing_certificate> (8)
  rules: (9)
  - operations: (10)
    - <operation>
    apiGroups:
    - ""
    apiVersions:
    - "*"
    resources:
    - <resource>
  failurePolicy: <policy> (11)
  sideEffects: Unknown
```

1. Spécifie la configuration d'un plugin de validation des admissions.
2. Le nom de l'objet **ValidatingWebhookConfiguration**. Remplacer **<webhook_name>** par la valeur appropriée.
3. Le nom du webhook à appeler. Remplacez **<webhook_name>** par la valeur appropriée.
4. Informations sur la manière de se connecter au serveur webhook, de lui faire confiance et de lui envoyer des données.
5. L'espace de noms dans lequel le service frontal est créé.
6. Le nom du service frontal.

7 URL du webhook utilisé pour les demandes d'admission. Remplacez **<webhook_url>** par la valeur appropriée.

8 Un certificat d'autorité de certification codé en PEM qui signe le certificat de serveur utilisé par le serveur webhook. Remplacez **<ca_signing_certificate>** par le certificat approprié au format base64.

9 Règles qui définissent quand le serveur API doit utiliser ce plugin d'admission au webhook.

10 Une ou plusieurs opérations qui déclenchent l'appel du serveur API à ce plugin d'admission au webhook. Les valeurs possibles sont **create**, **update**, **delete** ou **connect**. Remplacez **<operation>** et **<resource>** par les valeurs appropriées.

11 Indique comment la politique doit se dérouler si le serveur webhook n'est pas disponible. Remplacez **<policy>** par **Ignore** (pour accepter inconditionnellement la demande en cas d'échec) ou **Fail** (pour refuser la demande qui a échoué). L'utilisation de **Ignore** peut entraîner un comportement imprévisible pour tous les clients.

9.5. CONFIGURATION DE L'ADMISSION DYNAMIQUE

Cette procédure décrit les étapes de haut niveau de la configuration de l'admission dynamique. La fonctionnalité de la chaîne d'admission est étendue en configurant un plugin d'admission webhook pour appeler un serveur webhook.

Le serveur webhook est également configuré en tant que serveur API agrégé. Cela permet à d'autres composants d'OpenShift Container Platform de communiquer avec le webhook en utilisant des identifiants internes et facilite les tests à l'aide de la commande **oc**. En outre, cela permet un contrôle d'accès basé sur les rôles (RBAC) dans le webhook et empêche les informations sur les jetons provenant d'autres serveurs API d'être divulguées au webhook.

Conditions préalables

- Un compte OpenShift Container Platform avec un accès administrateur de cluster.
- L'OpenShift Container Platform CLI (**oc**) est installé.
- Une image de conteneur de serveur de webhook publiée.

Procédure

1. Construire une image de conteneur de serveur webhook et la mettre à la disposition du cluster à l'aide d'un registre d'images.
2. Créez une clé et un certificat d'autorité de certification locaux et utilisez-les pour signer la demande de signature de certificat (CSR) du serveur webhook.
3. Créer un nouveau projet pour les ressources webhook :

```
$ oc new-project my-webhook-namespace 1
```

1 Notez que le serveur webhook peut s'attendre à un nom spécifique.

4. Définir les règles RBAC pour le service API agrégé dans un fichier appelé **rbac.yaml**:

```
apiVersion: v1
kind: List
items:

- apiVersion: rbac.authorization.k8s.io/v1 (1)
  kind: ClusterRoleBinding
  metadata:
    name: auth-delegator-my-webhook-namespace
  roleRef:
    kind: ClusterRole
    apiGroup: rbac.authorization.k8s.io
    name: system:auth-delegator
  subjects:
  - kind: ServiceAccount
    namespace: my-webhook-namespace
    name: server

- apiVersion: rbac.authorization.k8s.io/v1 (2)
  kind: ClusterRole
  metadata:
    annotations:
    name: system:openshift:online:my-webhook-server
  rules:
  - apiGroups:
    - online.openshift.io
    resources:
    - namespacereservations (3)
    verbs:
    - get
    - list
    - watch

- apiVersion: rbac.authorization.k8s.io/v1 (4)
  kind: ClusterRole
  metadata:
    name: system:openshift:online:my-webhook-requester
  rules:
  - apiGroups:
    - admission.online.openshift.io
    resources:
    - namespacereservations (5)
    verbs:
    - create

- apiVersion: rbac.authorization.k8s.io/v1 (6)
  kind: ClusterRoleBinding
  metadata:
    name: my-webhook-server-my-webhook-namespace
  roleRef:
    kind: ClusterRole
    apiGroup: rbac.authorization.k8s.io
    name: system:openshift:online:my-webhook-server
  subjects:
  - kind: ServiceAccount
    namespace: my-webhook-namespace
```

```
    name: server

- apiVersion: rbac.authorization.k8s.io/v1 7
  kind: RoleBinding
  metadata:
    namespace: kube-system
    name: extension-server-authentication-reader-my-webhook-namespace
  roleRef:
    kind: Role
    apiGroup: rbac.authorization.k8s.io
    name: extension-apiserver-authentication-reader
  subjects:
  - kind: ServiceAccount
    namespace: my-webhook-namespace
    name: server

- apiVersion: rbac.authorization.k8s.io/v1 8
  kind: ClusterRole
  metadata:
    name: my-cluster-role
  rules:
  - apiGroups:
    - admissionregistration.k8s.io
    resources:
    - validatingwebhookconfigurations
    - mutatingwebhookconfigurations
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - ""
    resources:
    - namespaces
    verbs:
    - get
    - list
    - watch

- apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRoleBinding
  metadata:
    name: my-cluster-role
  roleRef:
    kind: ClusterRole
    apiGroup: rbac.authorization.k8s.io
    name: my-cluster-role
  subjects:
  - kind: ServiceAccount
    namespace: my-webhook-namespace
    name: server
```

1 Délégue l'authentification et l'autorisation à l'API du serveur webhook.

2 Permet au serveur webhook d'accéder aux ressources du cluster.

3. Pointe vers des ressources. Cet exemple pointe vers la ressource **namespacereservations**.
4. Permet au serveur API agrégé de créer des examens d'admission.
5. Pointe vers des ressources. Cet exemple pointe vers la ressource **namespacereservations**.
6. Permet au serveur webhook d'accéder aux ressources du cluster.
7. Liaison de rôles pour lire la configuration de la fin de l'authentification.
8. Rôle de cluster par défaut et liaisons de rôle de cluster pour un serveur d'API agrégé.

5. Appliquer ces règles RBAC au cluster :

```
$ oc auth reconcile -f rbac.yaml
```

6. Créer un fichier YAML appelé **webhook-daemonset.yaml** qui est utilisé pour déployer un webhook en tant que serveur daemon dans un espace de noms :

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  namespace: my-webhook-namespace
  name: server
  labels:
    server: "true"
spec:
  selector:
    matchLabels:
      server: "true"
  template:
    metadata:
      name: server
      labels:
        server: "true"
    spec:
      serviceAccountName: server
      containers:
      - name: my-webhook-container 1
        image: <image_registry_username>/<image_path>:<tag> 2
        imagePullPolicy: IfNotPresent
        command:
        - <container_commands> 3
        ports:
        - containerPort: 8443 4
        volumeMounts:
        - mountPath: /var/serving-cert
          name: serving-cert
        readinessProbe:
          httpGet:
            path: /healthz
            port: 8443 5
```

```
        scheme: HTTPS
    volumes:
    - name: serving-cert
      secret:
        defaultMode: 420
        secretName: server-serving-cert
```

1. Notez que le serveur webhook peut s'attendre à un nom de conteneur spécifique.
2. Pointe vers l'image d'un conteneur de serveur webhook. Remplacez **<image_registry_username>/<image_path>:<tag>** par la valeur appropriée.
3. Spécifie les commandes d'exécution du conteneur webhook. Remplacez **<container_commands>** par la valeur appropriée.
4. Définit le port cible au sein des pods. Cet exemple utilise le port 8443.
5. Spécifie le port utilisé par la sonde de préparation. Cet exemple utilise le port 8443.

7. Déployer le jeu de démons :

```
$ oc apply -f webhook-daemonset.yaml
```

8. Définir un secret pour le signataire du certificat du service, dans un fichier YAML appelé **webhook-secret.yaml**:

```
apiVersion: v1
kind: Secret
metadata:
  namespace: my-webhook-namespace
  name: server-serving-cert
type: kubernetes.io/tls
data:
  tls.crt: <server_certificate> 1
  tls.key: <server_key> 2
```

1. Fait référence au certificat signé du serveur webhook. Remplacez **<server_certificate>** par le certificat approprié au format base64.
2. Fait référence à la clé signée du serveur webhook. Remplacez **<server_key>** par la clé appropriée au format base64.

9. Créer le secret :

```
$ oc apply -f webhook-secret.yaml
```

10. Définir un compte de service et un service dans un fichier YAML appelé **webhook-service.yaml**:

```
apiVersion: v1
kind: List
items:
```

```
- apiVersion: v1
  kind: ServiceAccount
  metadata:
    namespace: my-webhook-namespace
    name: server

- apiVersion: v1
  kind: Service
  metadata:
    namespace: my-webhook-namespace
    name: server
    annotations:
      service.beta.openshift.io/serving-cert-secret-name: server-serving-cert
  spec:
    selector:
      server: "true"
    ports:
    - port: 443 (1)
      targetPort: 8443 (2)
```

(1) Définit le port sur lequel le service écoute. Cet exemple utilise le port 443.

(2) Définit le port cible au sein des pods vers lequel le service redirige les connexions. Cet exemple utilise le port 8443.

11. Exposer le serveur webhook au sein du cluster :

```
$ oc apply -f webhook-service.yaml
```

12. Définir une définition de ressource personnalisée pour le serveur webhook, dans un fichier appelé **webhook-crd.yaml**:

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: namespacereservations.online.openshift.io (1)
spec:
  group: online.openshift.io (2)
  version: v1alpha1 (3)
  scope: Cluster (4)
  names:
    plural: namespacereservations (5)
    singular: namespacereservation (6)
    kind: NamespaceReservation (7)
```

(1) Reflète les valeurs de **CustomResourceDefinition spec** et est au format **<plural>.<group>**. Cet exemple utilise la ressource **namespacereservations**.

(2) Nom du groupe de l'API REST.

(3) Nom de la version de l'API REST.

(4) Les valeurs acceptées sont **Namespaced** ou **Cluster**.

5 Nom pluriel à inclure dans l'URL.

6 Alias vu dans la sortie de **oc**.

7 La référence pour les manifestes de ressources.

13. Appliquer la définition de la ressource personnalisée :

```
$ oc apply -f webhook-crd.yaml
```

14. Configurez également le serveur webhook en tant que serveur d'API agrégé, dans un fichier appelé **webhook-api-service.yaml**:

```
apiVersion: apiregistration.k8s.io/v1beta1
kind: APIService
metadata:
  name: v1beta1.admission.online.openshift.io
spec:
  caBundle: <ca_signing_certificate> 1
  group: admission.online.openshift.io
  groupPriorityMinimum: 1000
  versionPriority: 15
  service:
    name: server
    namespace: my-webhook-namespace
  version: v1beta1
```

1 Un certificat d'autorité de certification codé en PEM qui signe le certificat de serveur utilisé par le serveur webhook. Remplacez **<ca_signing_certificate>** par le certificat approprié au format base64.

15. Déployer le service API agrégé :

```
$ oc apply -f webhook-api-service.yaml
```

16. Définissez la configuration du plugin d'admission du webhook dans un fichier appelé **webhook-config.yaml**. Cet exemple utilise le plugin d'admission de validation :

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingWebhookConfiguration
metadata:
  name: namespacereservations.admission.online.openshift.io 1
webhooks:
- name: namespacereservations.admission.online.openshift.io 2
  clientConfig:
    service: 3
      namespace: default
      name: kubernetes
      path: /apis/admission.online.openshift.io/v1beta1/namespacereservations 4
    caBundle: <ca_signing_certificate> 5
  rules:
  - operations:
```

```
    - CREATE
    apiGroups:
    - project.openshift.io
    apiVersions:
    - "*"
    resources:
    - projectrequests
  - operations:
    - CREATE
    apiGroups:
    - ""
    apiVersions:
    - "*"
    resources:
    - namespaces
  failurePolicy: Fail
```

1. Nom de l'objet **ValidatingWebhookConfiguration**. Cet exemple utilise la ressource **namespacereservations**.
2. Nom du webhook à appeler. Cet exemple utilise la ressource **namespacereservations**.
3. Permet d'accéder au serveur webhook via l'API agrégée.
4. URL du webhook utilisée pour les demandes d'admission. Cet exemple utilise la ressource **namespacereservation**.
5. Un certificat d'autorité de certification codé en PEM qui signe le certificat de serveur utilisé par le serveur webhook. Remplacez **<ca_signing_certificate>** par le certificat approprié au format base64.

17. Déployer le webhook :

```
$ oc apply -f webhook-config.yaml
```

18. Vérifiez que le webhook fonctionne comme prévu. Par exemple, si vous avez configuré l'admission dynamique pour réserver des espaces de noms spécifiques, confirmez que les demandes de création de ces espaces de noms sont rejetées et que les demandes de création d'espaces de noms non réservés aboutissent.

9.6. RESSOURCES COMPLÉMENTAIRES

- Limitation des ressources réseau personnalisées gérées par le plugin de périphérique réseau SR-IOV
- Définir des tolérances qui permettent aux taints de qualifier les pods qui doivent être programmés sur un nœud
- Validation de la classe de priorité du pod