



# Red Hat Training and Certification

## Manuel d'exercices (ROLE)

OCP 4.10 DO285

**Administration de conteneurs, de  
Kubernetes et de Red Hat OpenShift II**  
Édition 2







## Join a community dedicated to learning open source

The Red Hat® Learning Community is a collaborative platform for users to accelerate open source skill adoption while working with Red Hat products and experts.



**Network** with tens of thousands of community members



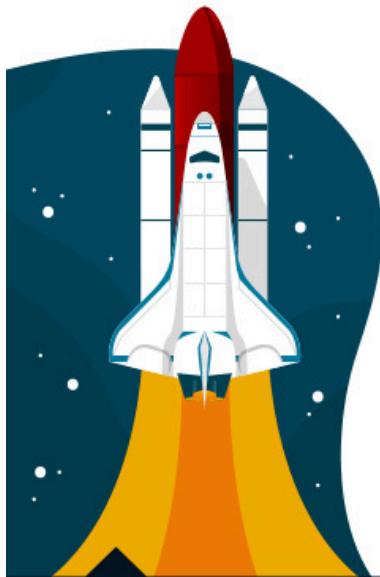
**Engage** in thousands of active conversations and posts



**Join and interact** with hundreds of certified training instructors



**Unlock** badges as you participate and accomplish new goals



This knowledge-sharing platform creates a space where learners can connect, ask questions, and collaborate with other open source practitioners.

**Access** free Red Hat training videos

**Discover** the latest Red Hat Training and Certification news

**Connect** with your instructor - and your classmates - before, after, and during your training course.

**Join** peers as you explore Red Hat products

Join the conversation [learn.redhat.com](https://learn.redhat.com)



Copyright © 2020 Red Hat, Inc. Red Hat, Red Hat Enterprise Linux, the Red Hat logo, and Ansible are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.



# **Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II**



# OCP 4.10 DO285

## Administration de conteneurs, de Kubernetes et de Red Hat

### OpenShift II

#### Édition 2 20220808

#### Date de publication 20220808

Auteurs: Richard Allred, Aykut Bulgu, Federico Fapitalle, Zach Guterman, Shatakshi Jain, Michael Jarrett, Dan Kolepp, Sourabh Mishra, Maria Fernanda Ordonez Casado, Eduardo Ramirez Ronco, Harpal Singh, Jordi Sola Alaball, Andres Hernandez, Ivan Chavero  
Éditeurs: Sam Ffrench, Seth Kenlon, Nicole Muller, Connie Petlitzer, Dave Sacco

Copyright © 2022 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are  
Copyright © 2022 Red Hat, Inc.

No part of this publication may be stored in a retrieval system, transmitted or reproduced in any way, including, but not limited to, photocopy, photograph, magnetic, electronic or other record, without the prior written permission of Red Hat, Inc.

This instructional program, including all material provided herein, is supplied without any guarantees from Red Hat, Inc. Red Hat, Inc. assumes no liability for damages or legal action arising from the use or misuse of contents or details contained herein.

If you believe Red Hat training materials are being used, copied, or otherwise improperly distributed, please send email to [training@redhat.com](mailto:training@redhat.com) or phone toll-free (USA) +1 (866) 626-2994 or +1 (919) 754-3700.

Red Hat, Red Hat Enterprise Linux, the Red Hat logo, JBoss, OpenShift, Fedora, Hibernate, Ansible, CloudForms, RHCA, RHCE, RHCSA, Ceph, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle American, Inc. and/or its affiliates.

XFS® is a registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is a trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack word mark and the Square O Design, together or apart, are trademarks or registered trademarks of OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. Red Hat, Inc. is not affiliated with, endorsed by, or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

Contributeurs : Forrest Taylor, Manuel Aude Morales, James Mighion, Michael Phillips, Fiona Allen

<b>Conventions de la documentation</b>	<b>xi</b>
	xi
<b>Introduction</b>	<b>xiii</b>
Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II .....	xiii
Organisation de l'environnement de la classe .....	xiv
Exécution des exercices pratiques .....	xxiii
<b>1. Présentation de la technologie de conteneur</b>	<b>1</b>
Aperçu de la technologie de conteneur .....	2
Quiz: Aperçu de la technologie de conteneur .....	5
Présentation de l'architecture de conteneur .....	9
Quiz: Présentation de l'architecture de conteneur .....	12
Aperçu de Kubernetes et OpenShift .....	14
Quiz: Description de Kubernetes et d'OpenShift .....	17
Exercice guidé: Configuration de l'environnement de formation .....	19
Résumé .....	27
<b>2. Crédation de services en conteneur</b>	<b>29</b>
Approvisionnement de services en conteneur .....	30
Exercice guidé: Crédation d'une instance de base de données MySQL .....	36
Résumé .....	39
<b>3. Gestion des conteneurs</b>	<b>41</b>
Gestion du cycle de vie des conteneurs .....	42
Exercice guidé: Gestion d'un conteneur MySQL .....	50
Association du stockage persistant à des conteneurs .....	54
Exercice guidé: Crédier un conteneur MySQL avec une base de données persistante .....	58
Accès aux conteneurs .....	61
Exercice guidé: Chargement de la base de données .....	63
Open Lab: Gestion des conteneurs .....	66
Résumé .....	76
<b>4. Gestion des images de conteneur</b>	<b>77</b>
Accès aux registres .....	78
Quiz: Utilisation des registres .....	85
Manipulation d'images de conteneur .....	89
Exercice guidé: Crédation d'une image de conteneur Apache personnalisée .....	95
Open Lab: Gestion des images .....	100
Résumé .....	109
<b>5. Crédation d'images de conteneur personnalisées</b>	<b>111</b>
Conception d'images de conteneur personnalisées .....	112
Quiz: Méthodes relatives à la conception d'images de conteneur .....	116
Compilation d'images de conteneur personnalisées avec des Containerfiles .....	118
Exercice guidé: Crédation d'une image de conteneur Apache de base .....	124
Open Lab: Crédation d'images de conteneur personnalisées .....	128
Résumé .....	135
<b>6. Déploiement d'applications en conteneur dans OpenShift</b>	<b>137</b>
Création de ressources Kubernetes .....	138
Exercice guidé: Déploiement d'un serveur de base de données dans OpenShift .....	151
Création de routes .....	156
Exercice guidé: Exposition d'un service en tant que route .....	160
Création d'applications avec Source-to-Image .....	165
Exercice guidé: Crédation d'une application en conteneur avec Source-to-Image .....	175
Open Lab: Déploiement d'applications en conteneur dans OpenShift .....	181
Résumé .....	186

<b>7. Déploiement d'applications multiconteneurs</b>	<b>187</b>
Déploiement d'une application multiconteneur sur OpenShift .....	188
Exercice guidé: Création d'une application sur OpenShift .....	190
Open Lab: Déploiement d'applications multiconteneurs .....	194
Résumé .....	201
<b>8. Description de Red Hat OpenShift Container Platform</b>	<b>203</b>
Description des fonctions de la plateforme OpenShift Container Platform .....	204
Quiz: Description des fonctions de la plateforme OpenShift Container Platform .....	209
Description de l'architecture d'OpenShift .....	213
Quiz: Description de l'architecture d'OpenShift .....	216
Description des opérateurs de cluster .....	220
Quiz: Description des opérateurs de cluster .....	223
Résumé .....	225
<b>9. Vérification de la santé d'un cluster</b>	<b>227</b>
Description des méthodes d'installation .....	228
Quiz: Description des méthodes d'installation .....	231
Résolution des problèmes relatifs aux applications et aux clusters OpenShift .....	233
Exercice guidé: Résolution des problèmes relatifs aux applications et aux clusters OpenShift .....	242
Présentation du stockage dynamique OpenShift .....	249
Exercice guidé: Présentation du stockage dynamique OpenShift .....	253
Résumé .....	258
<b>10. Configuration de l'authentification et de l'autorisation</b>	<b>259</b>
Configuration des fournisseurs d'identité .....	260
Exercice guidé: Configuration des fournisseurs d'identité .....	268
Définition et application des permissions à l'aide des RBAC .....	278
Exercice guidé: Définition et application des permissions à l'aide des RBAC .....	283
Open Lab: Configuration de l'authentification et de l'autorisation .....	289
Résumé .....	298
<b>11. Configuration de la sécurité des applications</b>	<b>299</b>
Gestion des informations sensibles à l'aide des secrets .....	300
Exercice guidé: Gestion des informations sensibles à l'aide des secrets .....	306
Contrôle des autorisations d'application à l'aide des contraintes du contexte de sécurité (SCC) .....	312
Exercice guidé: Contrôle des autorisations d'application à l'aide des contraintes du contexte de sécurité (SCC) .....	315
Open Lab: Configuration de la sécurité des applications .....	319
Résumé .....	327
<b>12. Configuration de la mise en réseau OpenShift pour les applications</b>	<b>329</b>
Résolution des problèmes liés au réseau défini par logiciel OpenShift .....	330
Exercice guidé: Résolution des problèmes liés au réseau défini par logiciel OpenShift .....	337
Rendre les applications accessibles en externe .....	346
Exercice guidé: Rendre les applications accessibles en externe .....	352
Configuration des politiques réseau .....	363
Exercice guidé: Configuration des politiques réseau .....	367
Open Lab: Configuration de la mise en réseau OpenShift pour les applications .....	376
Résumé .....	389
<b>13. Contrôle de la planification des pods</b>	<b>391</b>
Contrôle du comportement de planification des pods .....	392
Exercice guidé: Contrôle du comportement de planification des pods .....	399
Limitation de l'utilisation des ressources par une application .....	406
Exercice guidé: Limitation de l'utilisation des ressources par une application .....	417

Mise à l'échelle d'une application .....	427
Exercice guidé: Mise à l'échelle d'une application .....	431
Open Lab: Contrôle de la planification des pods .....	437
Résumé .....	445
<b>14. Description des mises à jour du cluster</b>	<b>447</b>
Description du processus de mise à jour du cluster .....	448
Quiz: Description du processus de mise à jour du cluster .....	460
Résumé .....	464
<b>15. Gestion d'un cluster avec la console Web</b>	<b>465</b>
Exécution de l'administration du cluster .....	466
Exercice guidé: Exécution de l'administration du cluster .....	470
Gestion des charges de travail et des opérateurs .....	477
Exercice guidé: Gestion des charges de travail et des opérateurs .....	482
Examen des mesures du cluster .....	486
Exercice guidé: Examen des mesures du cluster .....	490
Open Lab: Gestion d'un cluster avec la console Web .....	495
Résumé .....	505
<b>16. Révision complète</b>	<b>507</b>
Révision complète .....	508
Open Lab: Résoudre les problèmes relatifs aux applications et à un cluster OpenShift .....	510
Open Lab: Configurer un modèle de projet avec des restrictions de ressources et de réseau .....	527
<b>A. Création d'un compte GitHub</b>	<b>541</b>
Création d'un compte GitHub .....	542
<b>B. Création d'un compte Quay</b>	<b>545</b>
Création d'un compte Quay .....	546
Visibilité des référentiels .....	548
<b>C. Création d'un compte Red Hat</b>	<b>551</b>
Création d'un compte Red Hat .....	552
<b>D. Commandes Git utiles</b>	<b>555</b>
Commandes Git .....	556



# Conventions de la documentation

Cette section décrit les différentes conventions et pratiques utilisées dans tous les cours de la formation Red Hat.

## Avertissements

Les cours de la formation Red Hat font appel aux avertissements suivants :



### Références

Les références indiquent où trouver de la documentation externe se rapportant à un sujet.



### Note

Une « remarque » est un conseil, un raccourci ou une approche alternative pour la tâche en cours. Ignorer une remarque ne devrait pas entraîner de conséquences négatives, mais vous pourriez passer à côté d'une astuce qui vous simplifierait la vie.



### Important

Elles détaillent des éléments qui pourraient aisément être négligés : des changements de configuration qui ne s'appliquent qu'à la session en cours ou des services qui doivent être redémarrés pour qu'une mise à jour soit appliquée. Ignorer ces avertissements ne vous fera perdre aucune donnée, mais cela pourrait être source de frustration et d'irritation.



### Mise en garde

Un avertissement ne doit pas être ignoré. Le fait d'ignorer un avertissement risque fortement d'entraîner une perte de données.

## Langage inclusif

La formation Red Hat se penche actuellement sur les usages linguistiques dans différents domaines en vue d'éliminer tout emploi potentiellement offensant. Il s'agit d'un processus continu qui doit être appliqué aux produits et services couverts dans les cours de la formation Red Hat. Red Hat vous remercie pour votre patience tout au long du processus.



# Introduction

## Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II

Le cours « Containers, Kubernetes, and Red Hat OpenShift Administration II » (Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II) (DO285) vous permet d'acquérir des connaissances de base relatives à la création et à la gestion de conteneurs Linux et à Red Hat OpenShift Container Platform. Sur la base de travaux pratiques, vous découvrirez comment déployer des applications de test sur un exécutable de conteneur local ou un cluster OpenShift, et comment configurer et gérer des clusters OpenShift pour mieux comprendre comment les développeurs utiliseront la plateforme. Ces compétences sont nécessaires pour plusieurs rôles, notamment les développeurs, les administrateurs et les ingénieurs en fiabilité des sites.

### Objectifs du cours

- À la fin de ce cours, vous devez pouvoir démontrer les compétences nécessaires pour créer et gérer des conteneurs locaux à l'aide de Podman, établir un nouveau cluster OpenShift, effectuer la configuration initiale du cluster et gérer ce dernier au quotidien. L'un des objectifs principaux du cours consiste à résoudre les problèmes courants qui se posent au-delà de la première journée.

### Public

- Architectes système et logiciels souhaitant comprendre les fonctions et les fonctionnalités d'un cluster OpenShift. Administrateurs système souhaitant établir un cluster. Opérateurs de cluster souhaitant découvrir la maintenance continue d'un cluster. Ingénieurs en fiabilité des sites souhaitant découvrir la maintenance continue et la résolution des problèmes d'un cluster.

### Conditions préalables

- Obtenir l'examen RHCSA (Administrateur système certifié Red Hat) ou disposer de connaissances équivalentes.

# Organisation de l'environnement de la classe

Dans ce cours, le système informatique principal utilisé pour les travaux pratiques (exercices) est **workstation**.

La machine **workstation** possède un compte d'utilisateur standard, **student**, protégé par le mot de passe **student**. Aucun exercice de ce cours ne nécessite de vous connecter en tant que **root**, mais si vous le devez, le mot de passe **root** sur la machine **workstation** est **redhat**.

C'est à partir de la machine **workstation** que vous saisissez les commandes **oc** pour gérer le cluster OpenShift qui est préinstallé dans le cadre de votre environnement de formation.

C'est également à partir de la machine **workstation** que vous exécutez les scripts shell et les playbooks Ansible requis pour effectuer les exercices de ce cours.

Si les exercices nécessitent que vous ouvriez un navigateur Web pour accéder à une application ou à un site Web, vous devez alors utiliser la console graphique de la machine **workstation** et employer le navigateur Web Firefox à partir de là.



## Important

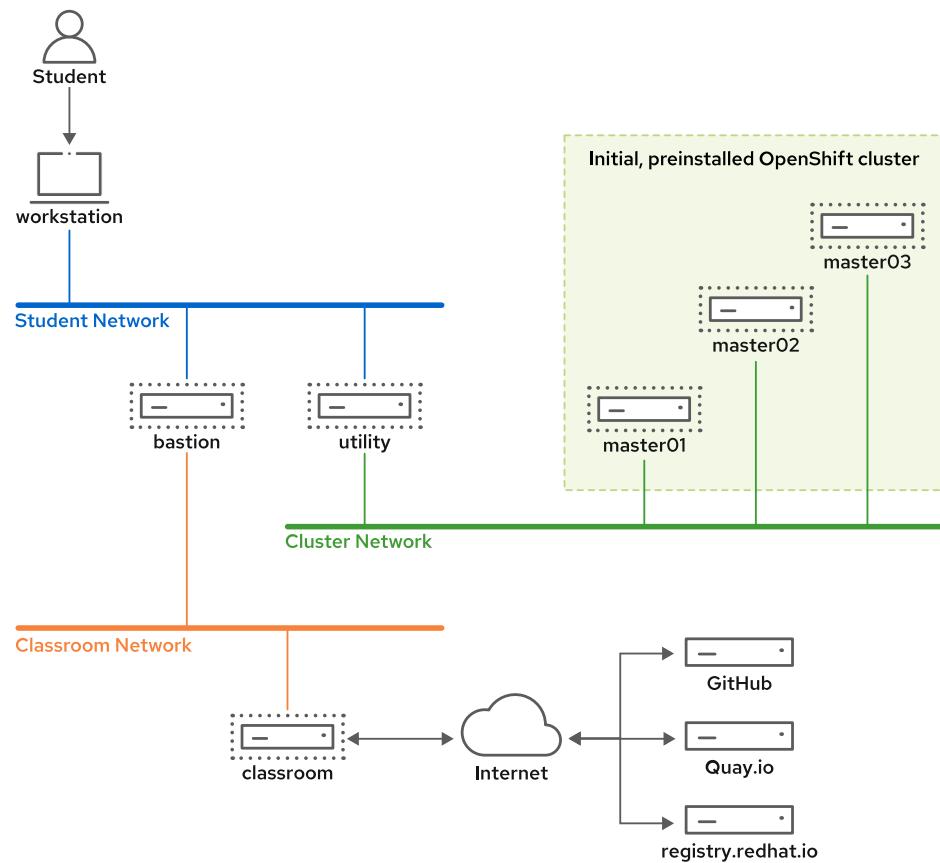
La première fois que vous démarrez votre environnement de formation, les clusters OpenShift mettent un peu plus de temps à devenir entièrement disponibles. La commande **lab** au début de chaque exercice effectue des vérifications et attend le cas échéant.

Si vous essayez d'accéder à votre cluster à l'aide de la commande **oc** ou de la console Web sans exécuter la commande **lab** au préalable, votre cluster ne sera peut-être pas encore disponible. Le cas échéant, attendez quelques minutes et réessayez.

Chaque stagiaire obtient un environnement de formation distant complet. Dans le cadre de cet environnement, chaque stagiaire obtient un cluster OpenShift dédié pour effectuer les tâches d'administration.

L'environnement de formation s'exécute entièrement comme des machines virtuelles dans un cluster Red Hat OpenStack Platform de grande taille qui est partagé entre de nombreux stagiaires.

La formation Red Hat met à disposition de nombreux clusters OpenStack, dans différents datacenters à travers le monde afin de réduire la latence pour les stagiaires de nombreux pays.



Toutes les machines présentes sur les réseaux Student, Classroom et Cluster exécutent Red Hat Enterprise Linux 8 (RHEL 8), à l'exception des machines qui sont des nœuds du cluster OpenShift. Ces dernières exécutent RHEL CoreOS.

Les systèmes appelés **bastion**, **utility** et **classroom** doivent toujours être en cours d'exécution. Ils fourniront les services d'infrastructure requis par l'environnement de formation et son cluster OpenShift. Il n'est pas prévu d'interagir directement avec ces systèmes.

Généralement, les commandes **lab** des exercices accèdent à ces machines lorsqu'il est nécessaire de configurer votre environnement pour l'exercice, et ne nécessitent aucune autre action de votre part.

Tous les systèmes du réseau **Student** se trouvent dans le domaine DNS `lab.example.com` et tous les systèmes du réseau **Classroom** se trouvent dans le domaine DNS `example.com`.

Les systèmes appelés **masterXX** sont des nœuds du cluster OpenShift 4 qui font partie de votre environnement de formation.

Tous les systèmes du réseau **de clusters** se trouvent dans le domaine DNS `ocp4.example.com`.

### Machines de la salle de classe

Nom de la machine	Adresses IP	RÔLE
<code>workstation.lab.example.com</code>	172.25.250.9	Poste de travail graphique utilisé pour l'administration du système

Nom de la machine	Adresses IP	RÔLE
classroom.example.com	172.25.254.254	Routeur reliant le réseau de la salle de classe à Internet
bastion.lab.example.com	172.25.250.254	Routeur reliant le réseau du stagiaire à celui de la salle de classe
utility.lab.example.com	172.25.250.253	Routeur reliant le réseau du stagiaire au réseau de clusters et également au serveur de stockage
master01.ocp4.example.com	192.168.50.10	Plan de contrôle et nœud de calcul
master02.ocp4.example.com	192.168.50.11	Plan de contrôle et nœud de calcul
master03.ocp4.example.com	192.168.50.12	Plan de contrôle et nœud de calcul

Red Hat OpenShift Container Platform 4 a besoin d'accéder à deux registres de conteneurs pour télécharger des images de conteneurs pour les opérateurs, les générateurs S2I et d'autres services de cluster. Ces registres sont les suivants :

- registry.redhat.io
- quay.io

Si l'un des deux registres n'est pas disponible lors du démarrage de l'environnement de formation, le cluster OpenShift peut ne pas démarrer ou entrer dans un état dégradé.

Si ces registres de conteneurs subissent une panne alors que l'environnement de formation est en cours d'exécution, il se peut qu'il ne soit pas possible d'effectuer des exercices tant que la panne n'est pas résolue.

Le cluster Red Hat OpenShift Container Platform 4 à l'intérieur de l'environnement de formation est préinstallé à l'aide de la méthode d'installation d'infrastructure préexistante ; tous les nœuds sont traités comme des serveurs nus, même s'ils sont en fait des machines virtuelles dans un cluster OpenStack.

Les fonctionnalités d'intégration du fournisseur de cloud OpenShift ne sont pas activées et certaines fonctionnalités qui dépendent de cette intégration, telles que les ensembles de machines et la mise à l'échelle automatique des nœuds de cluster, ne sont pas disponibles.

De nombreux exercices de la partie DO180 du cours requièrent l'exécution des tâches dans les exercices généraux du chapitre 1. Cela inclut l'exécution de la commande `lab-configure` et le clonage de votre branche du référentiel Git DO180-apps dans `/home/student/D0180-apps`. La commande `lab-configure` vous invite à saisir vos noms d'utilisateur GitHub et Quay.io, et enregistre les modifications apportées dans le fichier `/usr/local/etc/ocp4.config`.

Par défaut, le cluster OpenShift dans l'environnement de formation utilise le fournisseur d'identité HTPasswd et permet à l'utilisateur `developer` d'accéder à l'aide du mot de passe `developer`. Étant donné que la configuration d'OpenShift pour utiliser le fournisseur d'identité HTPasswd est un objectif de la partie DO280 du cours, certains scripts d'atelier suppriment la configuration du fournisseur d'identité. Si vous revenez à la partie DO180 du cours, exécutez la commande `lab-configure` pour vous assurer que le cluster OpenShift autorise l'accès à l'utilisateur `developer` à l'aide du mot de passe `developer`.

Si vous soupçonnez que vous ne pouvez pas vous connecter à votre cluster OpenShift en tant qu'utilisateur `admin`, car vous avez modifié les paramètres d'authentification de votre cluster de

## Introduction

manière incorrecte, exécutez la commande `lab finish` à partir de votre exercice en cours et redémarrez l'exercice en exécutant la commande `lab start`.

Pour les ateliers dans lesquels les utilisateurs `admin` et `developer` sont attendus, la commande `lab` réinitialise les paramètres d'authentification de cluster et restaure les mots de passe, de telle sorte que l'utilisateur `admin` ait le mot de passe `redhat` et que l'utilisateur `developer` ait le mot de passe `developer`.

Si l'exécution d'une commande `lab` ne permet pas de résoudre le problème, vous pouvez suivre les instructions de la section suivante pour utiliser la machine `utility` afin d'accéder à votre cluster OpenShift.

La machine `utility` a été utilisée pour exécuter le programme d'installation OpenShift à l'intérieur de votre environnement de formation, et il s'agit d'une ressource utile pour résoudre les problèmes de cluster. Vous pouvez afficher les manifestes du programme d'installation dans le dossier `/home/lab/ocp4` de la machine `utility`.

Il n'est pas nécessaire de se connecter au serveur `utility` pour effectuer des exercices. S'il semble que votre cluster OpenShift met trop de temps à démarrer ou qu'il se trouve dans un état dégradé, vous pouvez vous connecter à la machine `utility` en tant qu'utilisateur `lab` pour résoudre les problèmes de votre environnement de formation.

L'utilisateur `student` sur la machine `workstation` est déjà configuré avec des clés SSH qui lui permettent de se connecter à la machine `utility` sans mot de passe.

```
[student@workstation ~]$ ssh lab@utility
```

Dans la machine `utility`, l'utilisateur `lab` est préconfiguré avec un fichier `.kube/config` qui accorde l'accès en tant que `system:admin` sans nécessiter l'exécution préalable de la commande `oc login`.

Cela vous permet d'exécuter des commandes de résolution de problèmes, telles que `oc get node`, si elles échouent à partir de la machine `workstation`.

Vous ne devez pas exiger l'accès SSH à vos nœuds de cluster OpenShift pour les tâches d'administration normales, car OpenShift 4 fournit la commande `oc debug`; si nécessaire, l'utilisateur `lab` sur le serveur `utility` est préconfiguré avec des clés SSH pour accéder à tous les nœuds du cluster. Par exemple :

```
[lab@utility ~]$ ssh -i ~/.ssh/lab_rsa core@master01.ocp4.example.com
```

Dans l'exemple précédent, remplacez `master01` par le nom du nœud de cluster souhaité.

Les clusters Red Hat OpenShift Container Platform sont conçus pour fonctionner en continu, 24 h/24h, 7 j/7, jusqu'à ce qu'ils soient mis hors service. Contrairement à un cluster de production, l'environnement de formation contient un cluster qui a été arrêté après l'installation, et qui sera arrêté et redémarré quelques fois avant que vous ne terminiez ce cours. Cela présente un scénario qui nécessite une gestion spéciale qui n'est pas requise par un cluster de production.

Le plan de contrôle et les nœuds de calcul d'un cluster OpenShift communiquent fréquemment entre eux. Toutes les communications entre les nœuds du cluster sont protégées par une authentification mutuelle basée sur des certificats TLS par nœud.

Le programme d'installation d'OpenShift gère la création et l'approbation des demandes de signature de certificat TLS pour la méthode d'installation d'automatisation de la pile complète.

## Introduction

L'administrateur système doit approuver manuellement ces demandes de signature de certificat pour la méthode d'installation de l'infrastructure préexistante.

Tous les certificats TLS par nœud ont une durée d'expiration courte de 24 heures (la première fois) et de 30 jours (après renouvellement). Lorsqu'ils sont sur le point d'expirer, les nœuds de cluster concernés créent des demandes de signature de certificat et le plan de contrôle les approuve automatiquement. Si le plan de contrôle est hors ligne lorsque le certificat TLS d'un nœud expire, un administrateur de cluster doit approuver la demande de signature de certificat en attente.

La machine **utility** inclut un service système qui approuve les demandes de signature de certificat du cluster lors du démarrage du système de la salle de classe, afin de s'assurer que votre cluster est prêt au début des exercices. Si vous créez ou démarrez le système de la salle de classe et que vous commencez un exercice trop rapidement, il se peut que votre cluster ne soit pas encore prêt. Si tel est le cas, patientez quelques minutes pendant que la machine **utility** gère les demandes de signature de certificat, puis réessayez.

Parfois, la machine **utility** ne parvient pas à approuver toutes les demandes de signature de certificat requises, par exemple, parce que le cluster a mis trop de temps pour générer toutes les demandes de signature de certificat requises et que le service système n'a pas attendu suffisamment longtemps. Il est également possible que certains nœuds de cluster OpenShift n'ont pas suffisamment attendu l'approbation de leurs demandes de signature de certificat, et qu'ils ont émis de nouvelles demandes de signature de certificat qui ont remplacé les précédentes.

Si ces problèmes se produisent, vous remarquerez que votre cluster met trop de temps à réagir et que vos commandes `oc login` ou `lab` échouent. Pour résoudre le problème, vous pouvez vous connecter à la machine **utility**, comme indiqué précédemment, et exécuter le script `sign.sh` pour approuver les demandes de signature de certificat supplémentaires et en attente.

```
[lab@utility ~]$ ./sign.sh
```

Le script `sign.sh` exécute plusieurs boucles, au cas où vos nœuds de cluster émettent de nouvelles demandes de signature de certificat en remplacement de celles qu'il a approuvées.

Une fois que vous avez effectué l'approbation ou que le service système de la machine **utility** approuve toutes les demandes de signature de certificat, OpenShift doit redémarrer quelques opérateurs de cluster ; il faut quelques instants avant que votre cluster OpenShift soit prêt à répondre aux demandes des clients. Pour vous aider à gérer ce scénario, la machine **utility** fournit le script `wait.sh` qui attend que votre cluster OpenShift soit prêt à accepter les demandes d'authentification et d'API de la part des clients distants.

```
[lab@utility ~]$ ./wait.sh
```

Bien que cette situation soit improbable, si ni le service de la machine **utility** ni l'exécution des scripts `sigh.sh` et `wait.sh` rendent votre cluster OpenShift disponible pour commencer les exercices, ouvrez alors un ticket d'assistance client.

**Note**

Vous pouvez exécuter des commandes de résolution de problèmes à partir de la machine **utility** à tout moment, même si vous avez des nœuds de plan de contrôle qui ne sont pas prêts. Voici certaines commandes utiles :

**oc get nodes**

pour vérifier si tous les nœuds du cluster sont prêts.

**oc get csr**

pour vérifier si votre cluster a toujours des demandes de signature de certificat non autorisées en attente.

**oc get clusteroperators**

pour vérifier si l'un de vos opérateurs de cluster n'est pas disponible, est dans un état dégradé, ou s'il progresse dans la configuration et le déploiement de pods.

En cas d'échec, vous pouvez essayer de détruire et de recréer votre salle de classe en dernier recours avant de créer un ticket d'assistance client.

Les stagiaires se voient attribuer des ordinateurs distants dans une salle de classe de formation en ligne Red Hat. L'accès s'effectue par le biais d'une application web hébergée à l'adresse suivante : <https://rol.redhat.com/>. Pour se connecter à ce site, les stagiaires doivent utiliser leurs informations d'identification du Portail client Red Hat.

## Contrôle de vos systèmes

Des ordinateurs distants vous sont attribués au sein d'une salle de classe Red Hat Online Learning (ROLE). Vous avez accès à des cours d'autoformation par le biais d'une application Web hébergée à l'adresse suivante : [rol.redhat.com](http://rol.redhat.com) [<https://rol.redhat.com>]. Si votre cours est une formation virtuelle dispensée par un instructeur, vous recevrez l'URL de l'emplacement du cours. Connectez-vous à ce site à l'aide de vos informations d'identification du Portail Client Red Hat.

### Contrôle des machines virtuelles

Les machines virtuelles de votre environnement de formation sont contrôlées au moyen de commandes d'interface de page Web. L'état de chaque machine virtuelle est affiché sur l'onglet **Lab Environment**.

The screenshot shows the 'Lab Environment' tab selected in the top navigation bar. Below it, a section titled 'Lab Controls' contains buttons for 'DELETE' (red), 'STOP' (teal), and an information icon. A note says: 'Click CREATE to build all of the virtual machines needed for the classroom lab environment. This may take several minutes to complete. Once created the environment can then be stopped and restarted to pause your experience.' Another note says: 'If you DELETE your lab, you will remove all of the virtual machines in your classroom and lose all of your progress.' Below this is a table listing five virtual machines:

Machine Name	Status	Action	Open Console
bastion	active	ACTION	OPEN CONSOLE
classroom	active	ACTION	OPEN CONSOLE
servera	building	ACTION	OPEN CONSOLE
serverb	building	ACTION	OPEN CONSOLE
workstation	active	ACTION	OPEN CONSOLE

## États de la machine

État de la machine virtuelle	Description
BUILDING	La création initiale de la machine virtuelle est en cours.
ACTIVE	La machine virtuelle est en cours d'exécution et disponible. Si elle vient de démarrer, elle peut encore être en train de démarrer des services.
STOPPED	La machine virtuelle est complètement arrêtée. Au démarrage, la machine virtuelle affiche le même état qu'avant son arrêt. L'état du disque est préservé.

Selon l'état de la machine, une sélection des actions suivantes est disponible.

## Actions de la salle de classe

Bouton ou action	Description
CREATE	Permet de créer la salle de classe ROLE. Crée et démarre toutes les machines virtuelles nécessaires pour cette salle de classe.
CREATING	Les machines virtuelles de la salle de classe ROLE sont en cours de création. Crée et démarre toutes les machines virtuelles nécessaires pour cette salle de classe. Cette opération peut prendre plusieurs minutes.
DELETE	Permet de supprimer la salle de classe ROLE. Détruit toutes les machines virtuelles de la salle de classe. <b>Tous les travaux enregistrés sur les disques du système seront perdus.</b>
START	Permet de démarrer toutes les machines virtuelles de la salle de classe.

Bouton ou action	Description
STARTING	Toutes les machines virtuelles de la salle de classe démarrent.
STOP	Permet d'arrêter toutes les machines virtuelles de la salle de classe.

### Actions de la machine

Bouton ou action	Description
OPEN CONSOLE	Permet de se connecter à la console système de la machine virtuelle dans un nouvel onglet du navigateur. Vous pouvez vous connecter directement à la machine et exécuter des commandes, si nécessaire. Généralement, vous vous connectez uniquement à la machine virtuelle <b>workstation</b> , et à partir de là, vous utilisez ssh pour vous connecter aux autres machines virtuelles.
ACTION > Start	Permet de démarrer (allumer) la machine virtuelle.
ACTION > Shutdown	Permet d'éteindre correctement la machine virtuelle, en conservant le contenu du disque.
ACTION > Power Off	Force l'arrêt de la machine virtuelle, en conservant le contenu du disque. Cela équivaut à couper l'alimentation d'une machine physique.
ACTION > Reset	Force l'arrêt de la machine virtuelle et réinitialise le disque à son état initial. <b>Tous les travaux enregistrés sur le disque seront perdus.</b>

At the start of an exercise, if instructed to reset **a single virtual machine node**, click ACTION > Reset for only the specific virtual machine.

At the start of an exercise, if instructed to reset **all virtual machines**, click ACTION > Reset on every virtual machine in the list.

Si vous souhaitez que l'environnement de formation retourne à son état d'origine du début du cours, vous pouvez cliquer sur **DELETE** pour supprimer l'ensemble de l'environnement de formation. Une fois l'atelier supprimé, vous pouvez cliquer sur **CREATE** pour déployer un nouvel ensemble de systèmes de salle de classe.



#### Mise en garde

L'opération **DELETE** ne peut pas être annulée. **Tous les travaux que vous avez terminés dans l'environnement de formation seront perdus.**

### Minuterie d'arrêt automatique

L'inscription à la formation en ligne Red Hat vous alloue un temps machine fixe. Pour vous aider à préserver le temps qui vous est alloué, la salle de classe ROLE utilise des compteurs qui arrêtent ou suppriment la salle de classe à l'expiration du compteur approprié.

Pour régler ces compteurs, localisez les deux boutons + en bas de la page de gestion du cours. Cliquez sur le bouton d'arrêt automatique + pour ajouter une nouvelle heure au compteur d'arrêt automatique. Cliquez sur le bouton de destruction automatique + pour ajouter un nouveau jour au compteur de destruction automatique. Le maximum autorisé pour le bouton d'arrêt automatique

est de 12 heures et de 14 jours pour la destruction automatique. Veillez à ce que les compteurs restent définis pendant que vous travaillez, afin d'éviter que votre environnement ne s'arrête de manière inattendue. Veillez à ne pas régler les compteurs de façon inutilement élevée, ce qui pourrait gaspiller le temps d'abonnement qui vous a été alloué.

# Exécution des exercices pratiques

Exécutez la commande `lab` à partir de la machine `workstation` pour préparer votre environnement avant chaque exercice pratique, et exécutez-la à nouveau pour le nettoyer après un exercice. Chaque exercice pratique a un nom unique au sein d'un cours. L'exercice est précédé de `lab-` en tant que nom de fichier dans `/usr/local/lib`. Par exemple, l'exercice `instances-cli` a le nom de fichier `/usr/local/lib/lab-instances-cli`. Pour lister les exercices disponibles, utilisez la saisie semi-automatique par tabulation dans la commande `lab`. Notez que le mot « `tab` » dans la commande suivante fait référence à l'utilisation de la touche de tabulation de votre clavier :

```
[student@workstation ~]$ lab Tab Tab
administer-users  deploy-overcloud-lab  prep-deploy-ips          stacks-autoscale
analyze-metrics   instances-cli        prep-deploy-router      stacks-deploy
assign-roles       manage-interfaces  public-instance-deploy verify-overcloud
```

Il existe deux types d'exercices. Le premier type, l'exercice guidé, est un exercice pratique qui suit une explication du cours. Si une explication est suivie d'un quiz, cela indique généralement que le sujet ne comportait pas un exercice pratique réalisable. Le second type, les travaux pratiques en fin de chapitre, est un exercice noté pour vous aider à vérifier votre apprentissage. Lorsqu'un cours inclut une révision complète, les exercices de révision sont structurés sous la forme d'ateliers notés. La syntaxe d'exécution d'un script d'exercice est la suivante :

```
[student@workstation ~]$ lab exercise action
```

L'`action` est à choisir entre `start`, `grade` ou `finish`. Tous les exercices prennent en charge `start` et `finish`. Seuls les ateliers de fin de chapitre et de révision complète prennent en charge `grade`. Les cours plus anciens peuvent toujours utiliser `setup` et `cleanup` à la place des actions `start` et `finish` actuelles.

## `start`

Anciennement `setup`. La logique de démarrage du script vérifie les ressources requises pour commencer un exercice. Il peut s'agir notamment de configurer des paramètres, créer des ressources, vérifier les services prérequis et vérifier les résultats nécessaires des exercices précédents.

## `grade`

Les travaux pratiques en fin de chapitre permettent de vérifier ce que vous avez appris, après avoir effectué des exercices guidés précédents. L'action `grade` indique à la commande `lab` d'afficher une liste de critères de notation, avec un statut `PASS` ou `FAIL` pour chacun d'eux. Pour obtenir le statut `PASS` pour tous les critères, corrigez les erreurs et réexécutez l'action `grade`.

## `finish`

Anciennement `cleanup`. La logique de fin du script supprime les ressources de l'exercice qui ne sont plus nécessaires.

## Résolution des problèmes liés aux scripts de travaux pratiques

Les scripts d'exercice n'existent pas sur `workstation` tant qu'ils n'ont pas été exécutés pour la première fois. Lorsque vous exécutez la commande `lab` avec un exercice et une action valides, le script nommé `lab-exercise` est téléchargé du partage de contenu du serveur `classroom` vers `/usr/local/lib` sur la machine `workstation`. La commande `lab` crée deux fichiers journaux dans `/var/tmp/labs`, ainsi que le répertoire s'il n'existe pas encore. Un fichier nommé `exercise` capture les messages de sortie standard qui apparaissent normalement sur votre terminal. L'autre fichier, nommé `exercise.err`, capture les messages d'erreur.

```
[student@workstation ~]$ ls -l /usr/local/lib
-rwxr-xr-x. 1 root root 4131 May  9 23:38 lab-instances-cli
-rwxr-xr-x. 1 root root 93461 May  9 23:38 labtool.cl110.shlib
-rwxr-xr-x. 1 root root 10372 May  9 23:38 labtool.shlib

[student@workstation ~]$ ls -l /var/tmp/labs
-rw-r--r--. 1 root root   113 May  9 23:38 instances-cli
-rw-r--r--. 1 root root   113 May  9 23:38 instances-cli.err
```



### Note

Les scripts sont téléchargés à partir du partage `http://content.example.com/courses/COURSE/RELEASE/grading-scripts`, mais uniquement si le script n'existe pas encore sur la machine `workstation`.

Si vous devez télécharger à nouveau un script (par exemple, lorsqu'un script du partage est modifié), supprimez manuellement le script de l'exercice en cours de `/usr/local/lib` sur la machine `workstation`, puis exécutez à nouveau la commande `lab` pour l'exercice. Le script d'exercice plus récent est ensuite téléchargé à partir du partage `grading-scripts`.

Pour supprimer tous les scripts d'exercice en cours sur la machine `workstation`, utilisez la commande `lab` avec l'option `--refresh`. Une actualisation supprime tous les scripts de `/usr/local/lib`, mais ne supprime pas les fichiers journaux.

```
[student@workstation ~]$ lab --refresh
[student@workstation ~]$ ls -l /usr/local/lib

[student@workstation ~]$ ls -l /var/tmp/labs
-rw-r--r--. 1 root root   113 May  9 23:38 instances-cli
-rw-r--r--. 1 root root   113 May  9 23:38 instances-cli.err
```

## Interprétation des fichiers journaux de l'exercice

Les scripts d'exercice envoient des sorties dans les fichiers journaux, même lorsque les scripts sont correctement exécutés. Le texte d'en-tête de l'étape est ajouté entre les étapes, et les en-têtes de date et d'heure supplémentaires sont ajoutés au début de chaque exécution de script. Le journal de l'exercice contient normalement des messages qui indiquent l'exécution complète des étapes de commande. Par conséquent, le journal de résultat de l'exercice est utile pour observer les messages qui sont attendus si aucun problème n'est survenu, mais ne propose aucune aide supplémentaire en cas d'échec.

En revanche, le journal d'erreurs de l'exercice est plus utile pour la résolution des problèmes. Même lorsque les scripts sont correctement exécutés, les messages sont toujours envoyés au journal d'erreurs de l'exercice. Par exemple, un script qui vérifie qu'un objet existe déjà avant de tenter de le créer doit générer un message de type *object not found* lorsque l'objet n'existe pas encore. Dans ce scénario, ce message est attendu et n'indique aucun échec. Les messages d'échec réels sont généralement plus détaillés et les administrateurs système expérimentés doivent reconnaître les entrées courantes de message de journal.

Bien que les scripts d'exercice soient toujours exécutés à partir de la machine `workstation`, ils effectuent des tâches sur d'autres systèmes de l'environnement de cours. De nombreux environnements de cours, y compris OpenStack et OpenShift, utilisent une interface de ligne de commande (CLI) appelée à partir de la machine `workstation` pour communiquer avec des systèmes serveur à l'aide d'appels API. Étant donné que les actions de script répartissent généralement les tâches sur plusieurs systèmes, une résolution des problèmes supplémentaire est nécessaire pour déterminer l'emplacement d'une tâche ayant échoué. Connectez-vous à ces autres systèmes et utilisez les compétences de diagnostic Linux pour lire les fichiers journaux du système local et déterminer la cause de l'échec du script de travaux pratiques.

## Utilisation de Chromium comme alternative à Firefox

La machine `workstation` exécute Red Hat Enterprise Linux 8 (RHEL 8). Si vous rencontrez des problèmes avec le rendu des pages Web, le référentiel EPEL est activé sur la machine `workstation`. Vous pouvez donc installer le navigateur Web Chromium. Au besoin, exécutez la commande `sudo yum install chromium` pour installer Chromium.



## chapitre 1

# Présentation de la technologie de conteneur

### Objectif

Décrire la façon dont les logiciels peuvent s'exécuter dans des conteneurs orchestrés par Red Hat OpenShift Container Platform.

### Résultats

- Décrire la différence entre les applications conteneur et les déploiements traditionnels.
- Décrire les principes de base de l'architecture des conteneurs.
- Décrire les avantages de l'orchestration d'applications et d'OpenShift Container Platform.

### Sections

- Aperçu de la technologie de conteneur (avec quiz)
- Aperçu de l'architecture de conteneur (avec quiz)
- Aperçu de Kubernetes et OpenShift (avec quiz)

# Aperçu de la technologie de conteneur

---

## Résultats

Après avoir terminé cette section, les stagiaires seront en mesure de décrire la différence entre les applications en conteneur et les déploiements traditionnels.

## Applications en conteneur

Les applications logicielles dépendent généralement d'autres bibliothèques, fichiers de configuration ou services fournis par l'environnement d'exécution. L'environnement d'exécution traditionnel d'une application logicielle est un hôte physique ou une machine virtuelle, et les dépendances des applications sont installées en tant que partie intégrante de l'hôte.

Par exemple, considérons une application Python nécessitant l'accès à une bibliothèque partagée commune qui implémente le protocole TLS. En règle générale, un administrateur système installe le paquetage requis fournissant la bibliothèque partagée avant d'installer l'application Python.

L'inconvénient majeur d'une application logicielle déployée de manière traditionnelle est que les dépendances de l'application sont étroitement liées à l'environnement d'exécution.

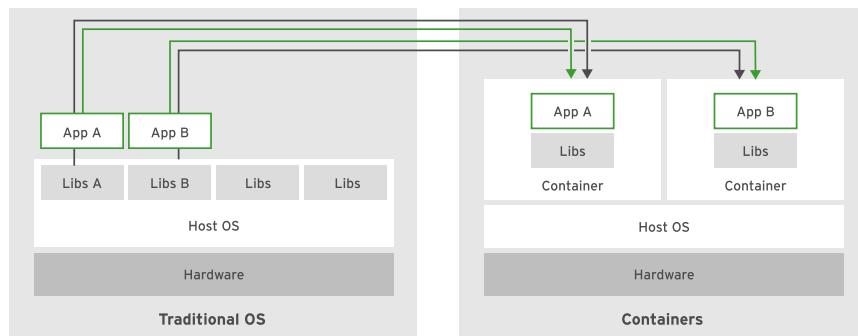
Une application peut tomber en panne lorsque des mises à jour ou des correctifs sont appliqués au système d'exploitation de base (OS).

Par exemple, une mise à jour de système d'exploitation vers la bibliothèque partagée TLS supprime TLS 1.0 en tant que protocole pris en charge. Ceci casse l'application Python déployée car elle est écrite pour utiliser le protocole TLS 1.0 pour les requêtes réseau. Cela force l'administrateur système à annuler la mise à jour du système d'exploitation pour maintenir l'exécution de l'application, empêchant ainsi les autres applications d'utiliser les avantages du package mis à jour.

Dès lors, une entreprise qui développe des logiciels applicatifs standard peut nécessiter une batterie complète de tests pour garantir que la mise à jour du système d'exploitation n'aura aucune incidence sur les applications exécutées sur l'hôte.

De plus, une application déployée de manière traditionnelle doit être arrêtée avant de mettre à jour les dépendances associées. Pour minimiser les temps d'arrêt des applications, les entreprises conçoivent et mettent en œuvre des systèmes complexes afin d'assurer la haute disponibilité de leurs applications. Conserver plusieurs applications sur un seul hôte devient souvent fastidieux et tout déploiement ou mise à jour peut potentiellement endommager l'une des applications de l'entreprise.

La *Figure 1.1* décrit la différence entre les applications s'exécutant en tant que conteneurs et les applications s'exécutant sur le système d'exploitation hôte.



**Figure 1.1: Différences entre un conteneur et un système d'exploitation**

Une application logicielle peut également être déployée à l'aide d'un conteneur.

Un conteneur est un ensemble d'un ou de plusieurs processus qui sont isolés du reste du système.

Les conteneurs fournissent nombre d'avantages similaires à ceux des machines virtuelles, tels que la sécurité, le stockage et l'isolement réseau. Les conteneurs nécessitent beaucoup moins de ressources matérielles et sont rapides à démarrer et à arrêter. Ils isolent également les bibliothèques et les ressources d'exécution (comme le processeur et l'espace de stockage) pour une application afin de minimiser l'impact d'une mise à jour du système d'exploitation sur le système d'exploitation hôte, comme illustré dans *Figure 1.1*.

L'utilisation des conteneurs améliore non seulement l'efficacité, la souplesse et la capacité de réutilisation des applications hébergées, mais aussi la portabilité des applications. L'OCI (Open Container Initiative) fournit un ensemble de normes de l'industrie qui définissent une spécification d'exécution de conteneur et une spécification d'image de conteneur. La spécification d'image définit le format de l'ensemble de fichiers et de métadonnées formant une image de conteneur. Lorsque vous créez une application en tant qu'image de conteneur, conforme à la norme OCI, vous pouvez utiliser n'importe quel moteur de conteneur compatible OCI pour exécuter l'application.

De nombreux moteurs de conteneurs permettent de gérer et d'exécuter des conteneurs individuels, notamment Rocket, Drawbridge, LXC, Docker et Podman. Podman est disponible dans Red Hat Enterprise Linux 7.6 et versions ultérieures, et est utilisé dans ce cours pour démarrer, gérer et terminer les conteneurs individuels.

Voici les avantages liés à l'utilisation des conteneurs :

### Faible encombrement du matériel

Les conteneurs utilisent des fonctionnalités internes du système d'exploitation pour créer un environnement isolé au sein duquel les ressources sont gérées au moyen de fonctions telles que des espaces de noms et des groupes de contrôle (cgroups). Cette méthode réduit la surcharge de mémoire et de processeur par rapport à un hyperviseur de machine virtuelle. Exécuter une application sur une machine virtuelle permet de l'isoler de l'environnement d'exécution. Cependant, cela exige une importante couche de services pour offrir le même degré d'encombrement réduit du matériel que celui fourni par les conteneurs.

### Isolement de l'environnement

Les conteneurs fonctionnent dans un environnement fermé, au sein duquel les modifications apportées au système d'exploitation hôte et aux autres applications n'ont pas d'incidence sur le conteneur. Étant donné que les bibliothèques dont un conteneur a besoin sont autonomes, l'application peut s'exécuter sans interruption. Par exemple, chaque application peut exister dans son propre conteneur avec son propre ensemble de bibliothèques. Une mise à jour apportée à un conteneur n'affecte pas les autres conteneurs.

## Déploiement rapide

Les conteneurs se déploient rapidement, car il n'est pas nécessaire d'installer tout le système d'exploitation sous-jacent. En règle générale, une nouvelle installation du système d'exploitation est nécessaire sur un hôte physique ou une machine virtuelle pour la prise en charge de l'isolement. De plus, une simple mise à jour peut nécessiter un redémarrage complet du système d'exploitation. Pour redémarrer un conteneur, il ne faut pas arrêter le moindre service sur le système d'exploitation hôte.

## Déploiement dans plusieurs environnements

Dans un scénario de déploiement classique à l'aide d'un seul hôte, toute différence d'environnement est susceptible d'interrompre l'application. Cependant, à l'aide de conteneurs, toutes les dépendances d'application et tous les paramètres d'environnement sont encapsulés dans l'image du conteneur.

## Capacité de réutilisation

Un même conteneur peut être réutilisé sans qu'il faille configurer un système d'exploitation complet. Par exemple, le même conteneur de base de données qui fournit un service de base de données de production peut être utilisé par chaque développeur pour créer une base de développement lors du développement d'applications. Grâce aux conteneurs, il n'est plus nécessaire de gérer des serveurs de base de données de production et de développement distincts. Une seule image de conteneur est utilisée pour créer des instances du service de base de données.

En règle générale, un logiciel applicatif et tous les services dépendants (bases de données, messagerie et systèmes de fichiers) sont censés s'exécuter dans un seul conteneur. Cela peut entraîner les mêmes problèmes associés aux déploiements de logiciels traditionnels sur des machines virtuelles ou des hôtes physiques. Dans ce cas, un déploiement à plusieurs conteneurs constitue peut-être une solution mieux adaptée.

En outre, les conteneurs constituent une approche idéale lorsque vous utilisez des microservices pour le développement d'applications. Chaque service est encapsulé dans un environnement de conteneur léger et fiable pouvant être déployé dans un environnement de production ou de développement. L'ensemble de services en conteneur requis par une application peuvent être hébergés sur un seul ordinateur, ce qui vous évite d'avoir à gérer un ordinateur pour chaque service.

En revanche, de nombreuses applications ne sont pas bien adaptées à un environnement en conteneur. Par exemple, les applications qui accèdent à des informations matérielles de bas niveau, telles que la mémoire, les systèmes de fichiers et les périphériques, peuvent s'avérer non fiables en raison de limitations liées au conteneur.



### Références

[Page d'accueil - OCI \(Open Container Initiative\)](#)

<https://www.opencontainers.org/>

## ► Quiz

# Aperçu de la technologie de conteneur

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- 1. **Parmi les options suivantes, citez deux exemples de logiciels applicatifs pouvant s'exécuter dans un conteneur. (Choisissez deux réponses.)**
- a. Une application Python contrôlée par une base de données qui accède à des services tels qu'une base de données MySQL, un serveur FTP (File Transfer Protocol) et un serveur Web sur un seul hôte physique.
  - b. Une application Java Enterprise Edition avec une base de données Oracle et un broker de messages s'exécutant sur une seule machine virtuelle.
  - c. Un outil de surveillance des E/S chargé de l'analyse du trafic et du transfert des données en bloc.
  - d. Une application de vidage de mémoire capable de prendre des instantanés à partir de tous les caches de processeur en mémoire à des fins de débogage.
- 2. **Parmi les cas d'utilisation cités ci-dessous, lesquels sont les mieux adaptés aux conteneurs ? (Choisissez deux réponses.)**
- a. Un fournisseur de logiciels doit distribuer un logiciel qui peut être réutilisé par d'autres entreprises rapidement et sans erreur.
  - b. Une entreprise qui déploie des applications sur un hôte physique souhaiterait améliorer ses performances en utilisant des conteneurs.
  - c. Les développeurs d'une entreprise ont besoin d'un environnement temporaire qui reproduise l'environnement de production afin de tester rapidement le code qu'ils développent.
  - d. Une société financière met en œuvre, sur ses propres conteneurs, un outil d'analyse des risques nécessitant une utilisation importante du processeur, dans le but de réduire le nombre de processeurs requis.

- 3. **Une entreprise fait migrer vers une nouvelle architecture ses applications PHP et Python qui s'exécutent sur le même hôte. En raison de politiques internes, toutes deux utilisent un ensemble de bibliothèques partagées personnalisées à partir du système d'exploitation. Cependant, la dernière mise à jour qui leur a été appliquée à la suite d'une demande de l'équipe de développement Python a entraîné l'interruption de l'application PHP. Quelles architectures offriront la meilleure prise en charge pour ces deux applications ? (Choisissez deux réponses.)**
- a. Déployer chaque application sur des machines virtuelles différentes et appliquer les bibliothèques partagées personnalisées séparément à chaque hôte de machine virtuelle.
  - b. Déployer chaque application sur des conteneurs différents et appliquer les bibliothèques partagées personnalisées séparément à chaque conteneur.
  - c. Déployer chaque application sur des machines virtuelles différentes et appliquer les bibliothèques partagées personnalisées à tous les hôtes de machine virtuelle.
  - d. Déployer chaque application sur des conteneurs différents et appliquer les bibliothèques partagées personnalisées à tous les conteneurs.
- 4. **Quels sont les types d'applications qui peuvent être empaquetés sous la forme de conteneurs en vue d'une utilisation immédiate ? (Choisissez trois réponses.)**
- a. Un hyperviseur de machine virtuelle
  - b. Un logiciel de blog, tel que WordPress
  - c. Une base de données
  - d. Un outil de récupération de système de fichiers local
  - e. Un serveur Web

## ► Solution

# Aperçu de la technologie de conteneur

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- 1. **Parmi les options suivantes, citez deux exemples de logiciels applicatifs pouvant s'exécuter dans un conteneur. (Choisissez deux réponses.)**
- a. Une application Python contrôlée par une base de données qui accède à des services tels qu'une base de données MySQL, un serveur FTP (File Transfer Protocol) et un serveur Web sur un seul hôte physique.
  - b. Une application Java Enterprise Edition avec une base de données Oracle et un broker de messages s'exécutant sur une seule machine virtuelle.
  - c. Un outil de surveillance des E/S chargé de l'analyse du trafic et du transfert des données en bloc.
  - d. Une application de vidage de mémoire capable de prendre des instantanés à partir de tous les caches de processeur en mémoire à des fins de débogage.
- 2. **Parmi les cas d'utilisation cités ci-dessous, lesquels sont les mieux adaptés aux conteneurs ? (Choisissez deux réponses.)**
- a. Un fournisseur de logiciels doit distribuer un logiciel qui peut être réutilisé par d'autres entreprises rapidement et sans erreur.
  - b. Une entreprise qui déploie des applications sur un hôte physique souhaiterait améliorer ses performances en utilisant des conteneurs.
  - c. Les développeurs d'une entreprise ont besoin d'un environnement temporaire qui reproduise l'environnement de production afin de tester rapidement le code qu'ils développent.
  - d. Une société financière met en œuvre, sur ses propres conteneurs, un outil d'analyse des risques nécessitant une utilisation importante du processeur, dans le but de réduire le nombre de processeurs requis.

- 3. **Une entreprise fait migrer vers une nouvelle architecture ses applications PHP et Python qui s'exécutent sur le même hôte. En raison de politiques internes, toutes deux utilisent un ensemble de bibliothèques partagées personnalisées à partir du système d'exploitation. Cependant, la dernière mise à jour qui leur a été appliquée à la suite d'une demande de l'équipe de développement Python a entraîné l'interruption de l'application PHP. Quelles architectures offriront la meilleure prise en charge pour ces deux applications ? (Choisissez deux réponses.)**
- a. Déployer chaque application sur des machines virtuelles différentes et appliquer les bibliothèques partagées personnalisées séparément à chaque hôte de machine virtuelle.
  - b. Déployer chaque application sur des conteneurs différents et appliquer les bibliothèques partagées personnalisées séparément à chaque conteneur.
  - c. Déployer chaque application sur des machines virtuelles différentes et appliquer les bibliothèques partagées personnalisées à tous les hôtes de machine virtuelle.
  - d. Déployer chaque application sur des conteneurs différents et appliquer les bibliothèques partagées personnalisées à tous les conteneurs.
- 4. **Quels sont les types d'applications qui peuvent être empaquetés sous la forme de conteneurs en vue d'une utilisation immédiate ? (Choisissez trois réponses.)**
- a. Un hyperviseur de machine virtuelle
  - b. Un logiciel de blog, tel que WordPress
  - c. Une base de données
  - d. Un outil de récupération de système de fichiers local
  - e. Un serveur Web

# Présentation de l'architecture de conteneur

---

## Résultats

Après avoir terminé cette section, vous devez pouvoir réaliser les tâches suivantes :

- Décrire l'architecture des conteneurs Linux.
- Décrire l'outil podman pour la gestion des conteneurs.

## Présentation de l'historique des conteneurs

Les conteneurs ont rapidement gagné en popularité ces dernières années. Cependant, la technologie derrière les conteneurs existe depuis assez longtemps. En 2001, Linux a présenté un projet appelé VServer. VServer a été la première tentative d'exécution d'ensembles complets de processus sur un serveur unique avec un degré élevé d'isolation.

À partir de VServer, le concept de processus isolés a encore évolué et s'est concrétisé autour des fonctionnalités suivantes du noyau Linux :

### Espaces de noms

Un espace de noms isole des ressources système spécifiques généralement visibles par tous les processus. À l'intérieur d'un espace de noms, seuls les processus qui en sont membres peuvent voir ces ressources. Les espaces de noms peuvent inclure des ressources telles que les interfaces réseau, la liste d'ID de processus, les points de montage, les ressources IPC et les informations de nom d'hôte du système.

### Groupes de contrôles (cGroups)

Les groupes de contrôle partitionnent les ensembles de processus et leurs enfants dans des groupes afin de gérer et de limiter les ressources qu'ils utilisent. Ces groupes imposent des restrictions quant à la quantité de ressources système utilisables par les processus. Ces restrictions empêchent un processus d'utiliser trop de ressources sur l'hôte.

### Seccomp

Développé en 2005 et introduit dans les conteneurs vers 2014, Seccomp limite la manière dont les processus peuvent utiliser les appels système. Seccomp définit un profil de sécurité pour les processus, qui liste les appels système, les paramètres et les descripteurs de fichiers qu'ils sont autorisés à utiliser.

### SELinux

SELinux (Security-Enhanced Linux) est un système de contrôle d'accès obligatoire pour les processus. Le noyau Linux utilise SELinux pour protéger les processus les uns des autres et pour protéger le système hôte des processus en cours. Les processus s'exécutent avec un type SELinux confiné qui possède un accès limité aux ressources du système hôte.

Toutes ces innovations et fonctionnalités reposent sur un concept de base : permettre l'exécution de processus isolés tout en continuant d'accéder aux ressources système. Ce concept constitue le socle de la technologie de conteneur et la base de toutes les implémentations de conteneur. De nos jours, les conteneurs sont des processus du noyau Linux utilisant ces fonctionnalités de sécurité pour créer un environnement isolé. Cet environnement interdit aux processus isolés d'utiliser de manière abusive le système ou d'autres ressources de conteneur.

## chapitre 1 | Présentation de la technologie de conteneur

Un cas d'utilisation courant de conteneurs consiste à avoir plusieurs répliques du même service (un serveur de base de données, par exemple) sur le même hôte. Chaque réplique comprend des ressources isolées (système de fichiers, ports, mémoire) ; il n'est donc pas nécessaire que le service gère le partage des ressources. L'isolement garantit qu'un service défectueux ou préjudiciable n'a pas d'incidence sur les autres services ou conteneurs du même hôte, ni sur le système sous-jacent.

## Description de l'architecture de conteneur Linux

Du point de vue du noyau Linux, un conteneur est un processus avec des restrictions. Cependant, au lieu d'exécuter un seul fichier binaire, un conteneur exécute une image. Une image est un ensemble de système de fichiers contenant toutes les dépendances nécessaires à l'exécution d'un processus : fichiers du système de fichiers, paquetages installés, ressources disponibles, processus en cours d'exécution et modules du noyau.

Tout comme les fichiers exécutables constituent la base de l'exécution des processus, les images constituent la base de l'exécution des conteneurs. Les conteneurs en cours d'exécution utilisent une vue immuable de l'image, ce qui permet à plusieurs conteneurs de réutiliser la même image simultanément. Les images étant des fichiers, elles peuvent être gérées par des systèmes de gestion de versions, ce qui améliore l'automatisation du conteneur et du déploiement d'images.

Les images de conteneur doivent être disponibles localement pour que le conteneur puisse les exécuter, mais les images sont généralement stockées et conservées dans un référentiel d'images. Un référentiel d'images est juste un service (public ou privé) où les images peuvent être stockées, recherchées et extraites. Les autres fonctionnalités fournies par les référentiels d'images sont l'accès à distance, les métadonnées d'image, et le contrôle des autorisations ou des versions d'image.

Il existe de nombreux référentiels d'images disponibles, chacun offrant des fonctionnalités différentes :

- Red Hat Container Catalog [<https://registry.redhat.io/>]
- Red Hat Quay [<https://quay.io/>]
- Docker Hub [<https://hub.docker.com/>]
- Registre de conteneurs Google [<https://cloud.google.com/container-registry/>]
- Registre de conteneurs Amazon Elastic [<https://aws.amazon.com/ecr/>]



### Note

Ce cours utilise le registre d'images publiques Quay, ce qui permet aux stagiaires de travailler avec des images sans se gêner les uns les autres.

## Gestion des conteneurs avec Podman

Les conteneurs, les images et les registres d'images doivent pouvoir interagir les uns avec les autres. Par exemple, vous devez être capable de créer des images et de les placer dans des registres d'images. Vous devez également pouvoir récupérer une image du registre d'images et créer un conteneur à partir de cette image.

Podman est un outil Open Source permettant de gérer les conteneurs et les images de conteneurs, et d'interagir avec les registres d'images. Il inclut les fonctions clés suivantes :

- Il utilise le format d'image spécifié par l'Open Container Initiative [<https://www.opencontainers.org/>] (OCI). Ces spécifications définissent un format d'image standard, communautaire et non propriétaire.

- Podman stocke les images locales dans un système de fichiers local. Cela évite d'avoir une architecture client/serveur inutile ou l'exécution de démons sur une machine locale.
- Podman suit les mêmes schémas de commande que l'interface de ligne de commande Docker. Inutile donc de vous familiariser avec un nouvel ensemble d'outils.
- Podman est compatible avec Kubernetes. Kubernetes peut utiliser Podman pour gérer ses conteneurs.



## Références

### **Registre de conteneurs Red Hat Quay**

<https://quay.io/>

### **Site Podman**

<https://podman.io/>

### **Open Container Initiative**

<https://www.opencontainers.org/>

## ► Quiz

# Présentation de l'architecture de conteneur

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les fonctionnalités Linux suivantes, lesquelles sont utilisées pour l'exécution de conteneurs ? (Choisissez trois réponses.)**
  - a. Espaces de noms
  - b. Gestion de l'intégrité
  - c. Security-Enhanced Linux
  - d. Groupes de contrôle
- ▶ 2. **Parmi les énoncés suivants, lequel décrit le mieux une image de conteneur ?**
  - a. Image de machine virtuelle à partir de laquelle un conteneur est créé.
  - b. Un ensemble de systèmes de fichiers contenant toutes les dépendances requises pour exécuter le processus à l'intérieur du conteneur.
  - c. Environnement d'exécution au sein duquel une application sera exécutée.
  - d. Fichier d'index du conteneur utilisé par un registre.
- ▶ 3. **Parmi les composants suivants, quels sont les trois qui sont communs aux implémentations d'architecture de conteneur ? (Choisissez trois réponses.)**
  - a. Exécutable de conteneur
  - b. Autorisations de conteneur
  - c. Images de conteneur
  - d. Registres de conteneurs
- ▶ 4. **Qu'est-ce qu'un conteneur par rapport au noyau Linux ?**
  - a. Machine virtuelle.
  - b. Un processus isolé avec un accès réglementé aux ressources.
  - c. Un ensemble de couches de système de fichiers exposées par UnionFS.
  - d. Un service externe fournissant des images de conteneur.
- ▶ 5. **Parmi les propositions suivantes, lesquelles sont des fonctionnalités de Podman ? (Choisissez-en deux.)**
  - a. Gérer les autorisations et la configuration du système d'exploitation pour exécuter des machines virtuelles.
  - b. Gérer des conteneurs, des images de conteneur et interagir avec des registres.
  - c. Exécuter un démon sur la machine locale pour exécuter des conteneurs.
  - d. Podman utilise les mêmes modèles de commande que Docker.

## ► Solution

# Présentation de l'architecture de conteneur

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les fonctionnalités Linux suivantes, lesquelles sont utilisées pour l'exécution de conteneurs ? (Choisissez trois réponses.)**
  - a. Espaces de noms
  - b. Gestion de l'intégrité
  - c. Security-Enhanced Linux
  - d. Groupes de contrôle
- ▶ 2. **Parmi les énoncés suivants, lequel décrit le mieux une image de conteneur ?**
  - a. Image de machine virtuelle à partir de laquelle un conteneur est créé.
  - b. Un ensemble de systèmes de fichiers contenant toutes les dépendances requises pour exécuter le processus à l'intérieur du conteneur.
  - c. Environnement d'exécution au sein duquel une application sera exécutée.
  - d. Fichier d'index du conteneur utilisé par un registre.
- ▶ 3. **Parmi les composants suivants, quels sont les trois qui sont communs aux implémentations d'architecture de conteneur ? (Choisissez trois réponses.)**
  - a. Exécutable de conteneur
  - b. Autorisations de conteneur
  - c. Images de conteneur
  - d. Registres de conteneurs
- ▶ 4. **Qu'est-ce qu'un conteneur par rapport au noyau Linux ?**
  - a. Machine virtuelle.
  - b. Un processus isolé avec un accès réglementé aux ressources.
  - c. Un ensemble de couches de système de fichiers exposées par UnionFS.
  - d. Un service externe fournissant des images de conteneur.
- ▶ 5. **Parmi les propositions suivantes, lesquelles sont des fonctionnalités de Podman ? (Choisissez-en deux.)**
  - a. Gérer les autorisations et la configuration du système d'exploitation pour exécuter des machines virtuelles.
  - b. Gérer des conteneurs, des images de conteneur et interagir avec des registres.
  - c. Exécuter un démon sur la machine locale pour exécuter des conteneurs.
  - d. Podman utilise les mêmes modèles de commande que Docker.

# Aperçu de Kubernetes et OpenShift

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Identifier les limitations des conteneurs Linux et le besoin d'orchestration de conteneur.
- Décrire l'outil d'orchestration de conteneur Kubernetes.
- Décrire Red Hat OpenShift Container Platform (RHOCOP).

## Limitations des conteneurs

Les conteneurs constituent un moyen simple d'empaqueter et d'exécuter des services. À mesure que le nombre de conteneurs gérés par une organisation augmente, le travail lié à leur démarrage manuel augmente de façon exponentielle, de pair avec la nécessité de répondre rapidement aux demandes externes.

Lorsqu'elles utilisent des conteneurs dans un environnement de production, les entreprises ont souvent besoin des éléments suivants :

- Une communication facile entre un grand nombre de services.
- Des limites de ressources sur les applications, quel que soit le nombre de conteneurs qui les exécutent.
- Des réponses aux pics d'utilisation des applications pour augmenter ou réduire les conteneurs en cours d'exécution.
- Une réaction à la détérioration des services.
- Un déploiement progressif d'une nouvelle version à un ensemble d'utilisateurs.

Les entreprises ont souvent besoin d'une technologie d'orchestration des conteneurs, car les exécutables de conteneurs (tels que Podman) ne répondent pas adéquatement aux exigences ci-dessus.

## Vue d'ensemble de Kubernetes

Kubernetes est un service d'orchestration qui simplifie le déploiement, la gestion et la mise à l'échelle des applications en conteneur.

La plus petite unité gérable dans Kubernetes est un pod. Un pod comprend un ou plusieurs conteneurs avec leurs ressources de stockage et une adresse IP qui représentent une seule application. Kubernetes utilise également les pods pour orchestrer les conteneurs qu'il contient et limiter ses ressources en une seule unité.

## Fonctions de Kubernetes

Kubernetes offre les fonctions suivantes en plus d'une infrastructure de conteneur :

## Découverte de service et équilibrage de charge

Kubernetes permet les communications entre services en affectant une entrée DNS unique à chaque ensemble de conteneurs. De cette manière, le service demandeur n'a besoin que de connaître le nom DNS de la cible, ce qui permet au cluster de modifier l'emplacement et l'adresse IP du conteneur, sans affecter le service. Cela permet d'équilibrer la charge de la demande sur le pool de conteneurs fournissant le service. Par exemple, Kubernetes peut diviser de manière égale les demandes entrantes vers un service MySQL en tenant compte de la disponibilité des pods.

## Mise à l'échelle horizontale

Les applications peuvent évoluer manuellement ou automatiquement avec une configuration définie, avec l'interface de ligne de commande Kubernetes ou l'interface utilisateur Web.

## Autoréparation

Kubernetes peut utiliser des bilans de santé définis par l'utilisateur pour surveiller les pods à redémarrer et les replanifier en cas d'échec.

## Déploiement automatisé

Kubernetes peut progressivement appliquer les mises à jour aux conteneurs de votre application tout en vérifiant leur statut. En cas de problème lors du déploiement, Kubernetes peut revenir à l'itération précédente du déploiement.

## Secrets et gestion de la configuration

Vous pouvez gérer les paramètres de configuration et les secrets de vos applications sans reconstruire les conteneurs. Les secrets d'application peuvent être des noms d'utilisateur, des mots de passe et des points d'accès de service, out tout autre paramètre de configuration devant rester confidentiel.

## Opérateurs

Les opérateurs sont des applications Kubernetes empaquetées qui apportent également la connaissance du cycle de vie de l'application dans le cluster Kubernetes. Les applications empaquetées en tant qu'opérateurs utilisent l'API Kubernetes pour mettre à jour l'état du cluster en fonction des modifications apportées à l'état de l'application.

# Aperçu d'OpenShift

Red Hat OpenShift Container Platform (RHOC) est un ensemble de composants modulaires et de services créés par-dessus une infrastructure de conteneur Kubernetes. RHOC ajoute les fonctionnalités permettant de fournir une plate-forme PaaS de production telles que la gestion à distance, une architecture multisite, une sécurité accrue, le contrôle et l'audit, la gestion du cycle de vie des applications et des interfaces en libre-service pour les développeurs.

À partir de Red Hat OpenShift v4, les hôtes d'un cluster OpenShift utilisent tous Red Hat Enterprise Linux CoreOS comme système d'exploitation sous-jacent.



### Note

Les termes RHOC et OpenShift sont utilisés dans le cadre de cette formation pour faire référence à Red Hat OpenShift Container Platform.

# Fonctions d'OpenShift

OpenShift ajoute les fonctions suivantes à un cluster Kubernetes :

### **Workflow de développement intégré**

RHOPC intègre un registre de conteneurs, des pipelines CI/CD et S2I , et un outil permettant de construire des artefacts à partir de référentiels sources en images de conteneur.

### **Routes**

Exposez facilement les services au monde extérieur.

### **Métriques et journalisation**

Incluez le service de métriques intégré et d'auto-analyse, et la journalisation agrégée.

### **Interface utilisateur unifiée**

OpenShift apporte des outils unifiés et une interface utilisateur permettant de gérer toutes les fonctionnalités.



#### **Références**

##### **Orchestration de conteneurs de qualité production - Kubernetes**

<https://kubernetes.io/>

##### **OpenShift: Container Application Platform by Red Hat, Built on Containers and Kubernetes (OpenShift : Container Application Platform par Red Hat, basé sur Docker et Kubernetes)**

<https://www.openshift.com/>

## ► Quiz

# Description de Kubernetes et d'OpenShift

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- 1. **Parmi les affirmations suivantes, lesquelles sont correctes concernant les limitations inhérentes aux conteneurs ? (Choisissez trois réponses.)**
- a. Les conteneurs sont facilement orchestrés en grand nombre.
  - b. Le manque d'automatisation augmente le temps de réponse en cas de problèmes.
  - c. Les conteneurs ne gèrent pas les échecs d'application internes.
  - d. Il n'y a pas d'équilibrage de charge pour les conteneurs.
  - e. Les conteneurs sont des applications empaquetées qui sont fortement isolées.
- 2. **Parmi les affirmations suivantes, lesquelles sont correctes concernant Kubernetes ? (Choisissez deux réponses.)**
- a. Kubernetes est un conteneur.
  - b. Kubernetes ne peut utiliser que des conteneurs Docker.
  - c. Kubernetes est un système d'orchestration de conteneur.
  - d. Kubernetes simplifie la gestion, le déploiement et la mise à l'échelle des applications en conteneurs.
  - e. Les applications gérées dans un cluster Kubernetes sont plus difficiles à gérer.
- 3. **Parmi les affirmations suivantes, lesquelles sont correctes concernant Red Hat OpenShift v4 ? (Choisissez trois réponses.)**
- a. OpenShift fournit des fonctions supplémentaires à une infrastructure Kubernetes.
  - b. Kubernetes et OpenShift s'excluent mutuellement.
  - c. Les hôtes OpenShift utilisent Red Hat Enterprise Linux comme système d'exploitation de base.
  - d. OpenShift simplifie le développement en incorporant une technologie Source-to-Image et des pipelines CI/CD.
  - e. OpenShift simplifie le routage et l'équilibrage de la charge.
- 4. **Quelles fonctionnalités OpenShift propose-t-il pour étendre les fonctionnalités Kubernetes ? (Choisissez deux réponses.)**
- a. Opérateurs et Operator Framework.
  - b. Routes pour exposer les services au monde extérieur.
  - c. Workflow de développement intégré.
  - d. Réparation automatique et bilans de santé.

## ► Solution

# Description de Kubernetes et d'OpenShift

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les affirmations suivantes, lesquelles sont correctes concernant les limitations inhérentes aux conteneurs ? (Choisissez trois réponses.)**
  - a. Les conteneurs sont facilement orchestrés en grand nombre.
  - b. Le manque d'automatisation augmente le temps de réponse en cas de problèmes.
  - c. Les conteneurs ne gèrent pas les échecs d'application internes.
  - d. Il n'y a pas d'équilibrage de charge pour les conteneurs.
  - e. Les conteneurs sont des applications empaquetées qui sont fortement isolées.
- ▶ 2. **Parmi les affirmations suivantes, lesquelles sont correctes concernant Kubernetes ? (Choisissez deux réponses.)**
  - a. Kubernetes est un conteneur.
  - b. Kubernetes ne peut utiliser que des conteneurs Docker.
  - c. Kubernetes est un système d'orchestration de conteneur.
  - d. Kubernetes simplifie la gestion, le déploiement et la mise à l'échelle des applications en conteneurs.
  - e. Les applications gérées dans un cluster Kubernetes sont plus difficiles à gérer.
- ▶ 3. **Parmi les affirmations suivantes, lesquelles sont correctes concernant Red Hat OpenShift v4 ? (Choisissez trois réponses.)**
  - a. OpenShift fournit des fonctions supplémentaires à une infrastructure Kubernetes.
  - b. Kubernetes et OpenShift s'excluent mutuellement.
  - c. Les hôtes OpenShift utilisent Red Hat Enterprise Linux comme système d'exploitation de base.
  - d. OpenShift simplifie le développement en incorporant une technologie Source-to-Image et des pipelines CI/CD.
  - e. OpenShift simplifie le routage et l'équilibrage de la charge.
- ▶ 4. **Quelles fonctionnalités OpenShift propose-t-il pour étendre les fonctionnalités Kubernetes ? (Choisissez deux réponses.)**
  - a. Opérateurs et Operator Framework.
  - b. Routes pour exposer les services au monde extérieur.
  - c. Workflow de développement intégré.
  - d. Réparation automatique et bilans de santé.

## ► Exercice guidé

# Configuration de l'environnement de formation

Au cours de cet exercice, vous allez configurer workstation pour qu'il accède à l'ensemble de l'infrastructure utilisée par ce cours.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Configurer votre machine workstation pour qu'elle accède à un cluster OpenShift, un registre d'images de conteneur et un référentiel Git utilisé tout au long du cours.
- Scinder le référentiel d'exemples d'applications de ce cours sur votre compte GitHub personnel.
- Cloner le référentiel d'exemples d'applications de ce cours à partir de votre compte GitHub personnel vers la machine workstation.

## Avant De Commencer

Pour effectuer cet exercice, assurez-vous d'avoir accès aux éléments suivants :

- Un accès au cours DO180 dans l'environnement de formation en ligne de Red Hat.
- Les paramètres de connexion et un compte d'utilisateur developer permettant d'accéder au cluster OpenShift fourni dans votre environnement de formation.
- Un compte personnel et gratuit GitHub. Si vous devez vous inscrire à GitHub, reportez-vous aux instructions dans *Annexe A, Création d'un compte GitHub*.
- Un compte personnel et gratuit Quay.io. Si vous devez vous inscrire à Quay.io, reportez-vous aux instructions dans *Annexe B, Création d'un compte Quay*.
- Un token d'accès personnel, GitHub.

## Instructions

Avant de commencer un exercice, veillez à effectuer les tâches suivantes :

- 1. Préparez votre token d'accès GitHub.
- 1.1. Accédez à <https://github.com> à l'aide d'un navigateur Web et authentifiez-vous.
  - 1.2. En haut de la page, cliquez sur votre icône de profil, puis sélectionnez le menu **Settings**.

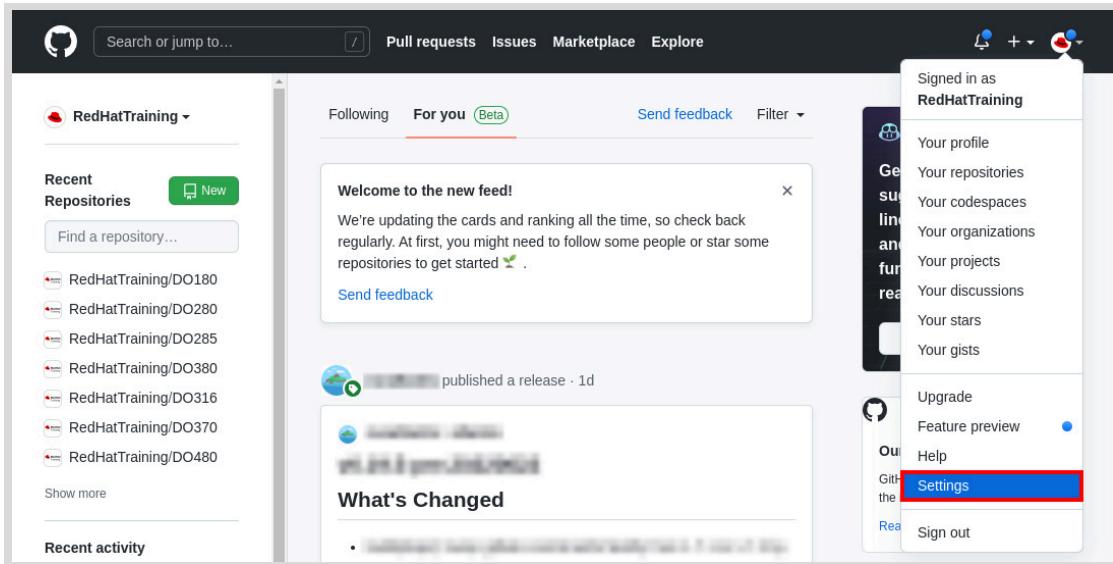


Figure 1.2: Menu User

Sélectionnez **Developer settings** dans le volet gauche de la page.

A screenshot of the GitHub developer settings page. The left sidebar lists various settings categories: Profile, Account, Appearance (with a 'New' badge), Account security, Billing &amp; plans, Security log, Security &amp; analysis, Emails, Notifications, SSH and GPG keys, Repositories, Packages, Organizations, Saved replies, Applications, and Developer settings. The 'Developer settings' option is highlighted with a red box. The main content area contains fields for Name, Public email, Bio, URL, Twitter username, Company, and a note about linking to a company organization. A profile picture placeholder with a checkered pattern is shown.

Figure 1.3: Paramètres Developer

- 1.3. Sélectionnez la section Personal access token dans le volet de gauche. Sur la page suivante, créez votre token en cliquant sur **Generate new token**. Vous êtes alors invité à entrer votre mot de passe.

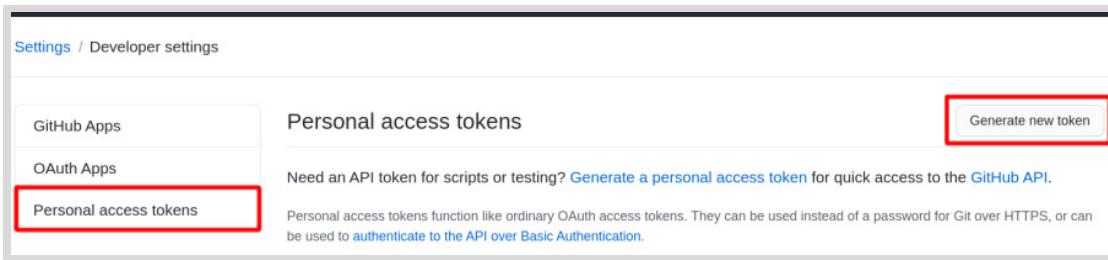


Figure 1.4: Volet Personal access token

- 1.4. Écrivez une brève description de votre nouveau token d'accès dans le champ Note.
- 1.5. Sélectionnez l'option public\_repo et ne décochez pas les autres options. Créez votre token d'accès en cliquant sur Generate token.

The screenshot shows the 'New personal access token' configuration page. It has a note field containing 'Course DO180' and a 'Select scopes' section. In the scopes section, the 'public\_repo' option is checked, while others like 'repo', 'repo:status', 'repo\_deployment', 'repo:invite', and 'security\_events' are unchecked. A red box highlights the 'public\_repo' checkbox.

Scopes	Description
<input type="checkbox"/> repo	Full control of private repositories
<input type="checkbox"/> repo:status	Access commit status
<input type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input type="checkbox"/> repo:invite	Access repository invitations
<input type="checkbox"/> security_events	Read and write security events

Figure 1.5: Configuration personnelle des tokens d'accès

- 1.6. Votre nouveau token d'accès personnel est affiché dans la sortie. À l'aide de l'éditeur de texte de votre choix, créez un fichier dans le répertoire personnel du stagiaire nommé token et assurez-vous de coller votre token d'accès personnel généré. Le token d'accès personnel ne peut plus être affiché dans GitHub.

The screenshot shows the GitHub 'Personal access tokens' page. At the top, there are buttons for 'Generate new token' and 'Revoke all'. Below this, a message says 'Tokens you have generated that can be used to access the [GitHub API](#)'. A note below says 'Make sure to copy your new personal access token now. You won't be able to see it again!'. A red box highlights a specific token entry: '✓ ghp\_kgYGzWcGE1CrdovkzuzeLTWvYY6eBX2l0vCK' with a copy icon, and a 'Delete' button to its right. Below the token list, a note states: 'Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API](#) over Basic Authentication'.

**Figure 1.6: Token d'accès généré**

- 1.7. Sur `workstation`, exécutez la commande `git config` avec les paramètres `credential.helper cache` pour stocker dans la mémoire cache vos informations d'identification en vue d'une utilisation ultérieure. Le paramètre `--global` applique la configuration à tous vos référentiels.

```
[student@workstation ~]$ git config --global credential.helper cache
```



#### Note

Le délai d'expiration par défaut est de 900 secondes. Vous pouvez en spécifier un autre si nécessaire.

```
[student@workstation ~]$ git config --global credential.helper \
  'cache --timeout=3600'
```



#### Important

Pendant ce cours, si vous êtes invité à saisir un mot de passe lors de l'utilisation d'opérations Git sur la ligne de commande, utilisez votre token d'accès comme mot de passe.

#### ► 2. Préparez votre mot de passe Quay.io.

- 2.1. Configurez un mot de passe pour votre compte Quay.io. Sur la page *Account Settings*, cliquez sur le lien *Change password*. Consultez Annexe B, *Création d'un compte Quay* pour plus de détails.

#### ► 3. Configurez la machine `workstation`.

Ouvrez un terminal sur la machine virtuelle `workstation` et exécutez la commande suivante.

```
[student@workstation ~]$ lab-configure
```

Répondez aux invites interactives avant de commencer tout autre exercice de ce cours.



### Note

Si vous faites une erreur, vous pouvez interrompre la commande à tout moment à l'aide de **Ctrl+C** et recommencer en ajoutant l'option **-d**.

```
[student@workstation ~]$ lab-configure -d
```

- 3.1. La commande **lab-configure** commence par afficher une série d'invites interactives et utilise des valeurs par défaut sensibles lorsqu'elles sont disponibles.

```
[student@workstation ~]$ lab-configure
```

- Enter the GitHub account name: **yourgituser** ①  
Verifying GitHub account name: **yourgituser**
  - Enter the Quay.io account name: **yourquayuser** ②  
Verifying Quay.io account name: **yourquayuser**
- ...output omitted...

**① ②** Vos noms de comptes GitHub et Quay.io personnels. Vous avez besoin de comptes gratuits et valides sur ces services en ligne pour effectuer les exercices de ce cours. Si vous n'avez jamais utilisé ces services en ligne, reportez-vous à *Annexe A, Création d'un compte GitHub* et *Annexe B, Création d'un compte Quay* pour obtenir des instructions sur la manière de procéder à l'inscription.

- 3.2. Si **lab-configure** trouve des problèmes, un message d'erreur s'affiche et l'application se ferme. Vous devez vérifier vos informations et exécuter à nouveau la commande **lab-configure -d**. La liste suivante montre un exemple d'erreur de vérification.

```
· Enter the Quay.io account name: notexists
Verifying Quay.io account name: notexists
ERROR: Cannot find Quay.io account user 'notexists'

To reconfigure, run: lab-configure -d
```

- 3.3. Pour finir, la commande **lab-configure** permet de s'assurer que l'utilisateur **developer** peut se connecter au cluster OpenShift.

```
...output omitted...

. Ensuring user 'developer' can log in to the OpenShift cluster.

...output omitted...
```

- 3.4. Si tous les contrôles réussissent, la commande **lab-configure** enregistre votre configuration :

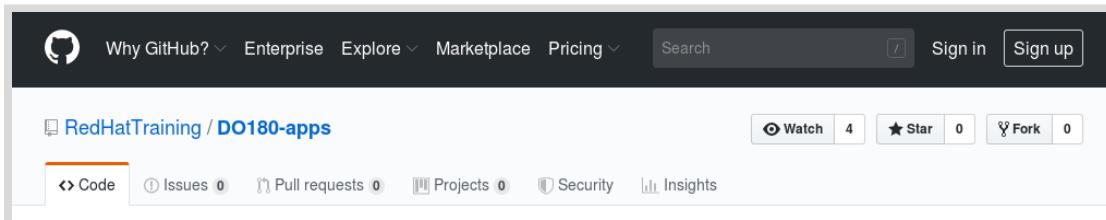
- 3.5. Si aucune erreur n'est survenue lors de l'enregistrement de votre configuration, vous êtes presque prêt à lancer l'un des exercices de ce cours. En cas d'erreur, n'essayez pas de démarrer un exercice tant que vous ne pouvez pas exécuter la commande `lab-configure` avec succès.

- 4. Scindez les exemples d'applications de ce cours dans votre compte GitHub personnel. Effectuez les étapes suivantes :

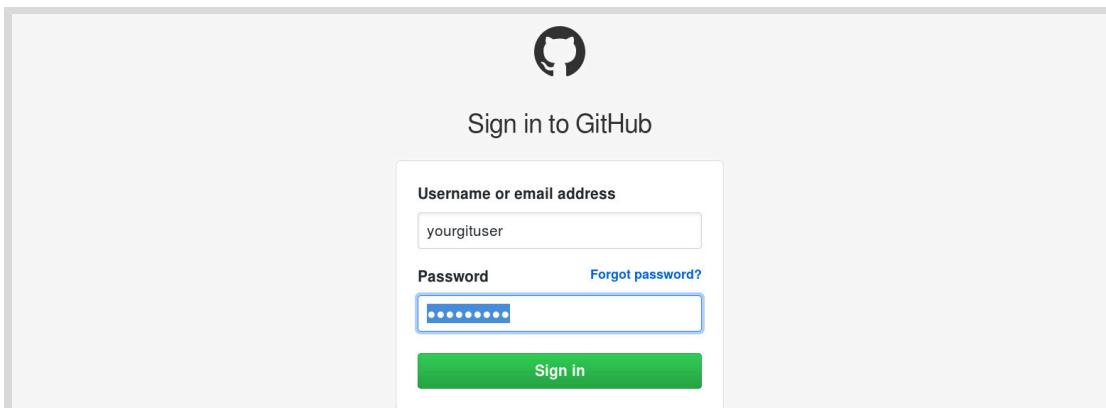
- 4.1. Ouvrez un navigateur Web et accédez au référentiel GitHub DO180-apps.

• <https://github.com/RedHatTraining/DO180-apps>

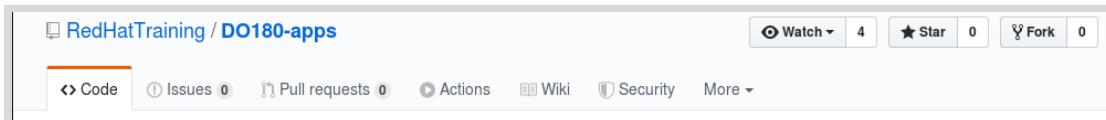
Si vous n'êtes pas connecté à GitHub, cliquez sur **Sign in** dans le coin supérieur droit.



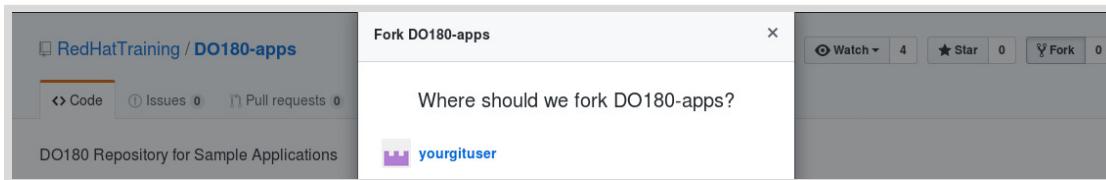
- 4.2. Connectez-vous à GitHub à l'aide de votre nom d'utilisateur personnel et de votre mot de passe.



- 4.3. Accédez au référentiel RedHatTraining/D0180-apps, puis cliquez sur **Fork** dans le coin supérieur droit.



- 4.4. Dans la fenêtre Fork D0180-apps, cliquez sur `yourgituser` pour sélectionner votre projet GitHub personnel.

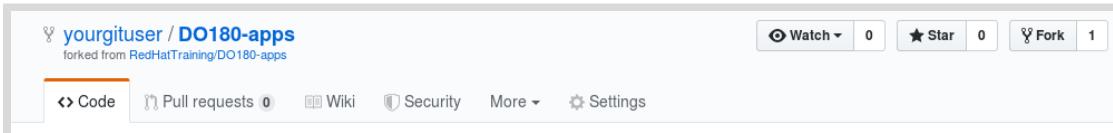




### Important

Bien qu'il soit possible de renommer votre branche personnelle du référentiel <https://github.com/RedHatTraining/DO180-apps>, les scripts de notation, les scripts d'aide et la sortie d'exemple de ce cours supposent que vous conservez le nom **D0180-apps** lorsque vous scindez le référentiel.

- 4.5. Après quelques minutes, l'interface Web de GitHub affiche votre nouveau référentiel *yourgituser/DO180-apps*.



- 5. Clonez les exemples d'applications de ce cours à partir de votre compte GitHub personnel vers votre machine **workstation**. Effectuez les étapes suivantes :

- 5.1. Exécutez la commande suivante pour cloner le référentiel d'exemples d'applications de ce cours. Remplacez *yourgituser* par le nom de votre compte GitHub personnel.

```
[student@workstation ~]$ git clone https://github.com/yourgituser/DO180-apps
cloning into 'DO180-apps'...
...output omitted...
```

- 5.2. Vérifiez que */home/student/DO180-apps* est un référentiel Git.

```
[student@workstation ~]$ cd DO180-apps
[student@workstation DO180-apps]$ git status
# On branch master
nothing to commit, working directory clean
```

- 5.3. Créez une branche pour tester votre nouveau token d'accès personnel.

```
[student@workstation DO180-apps]$ git checkout -b testbranch
Switched to a new branch testbranch
```

- 5.4. Modifiez le fichier TEST, puis validez-le sur Git.

```
[student@workstation DO180-apps]$ echo "D0180" > TEST
[student@workstation DO180-apps]$ git add .
[student@workstation DO180-apps]$ git commit -m "D0180"
...output omitted...
```

- 5.5. Transmettez par push les modifications à votre branche testing récemment créée.

```
[student@workstation DO180-apps]$ git push --set-upstream origin testbranch
Username for https://github.com: ①
Password for https://yourgituser@github.com: ②
...output omitted...
```

- ➊ Saisissez votre nom d'utilisateur GitHub.
  - ➋ Saisissez votre token d'accès personnel.
- 5.6. Apportez d'autres modifications à un fichier texte, validez-les et poussez-les. Vous remarquerez que vous n'êtes plus invité à fournir votre utilisateur et votre mot de passe. Cela est dû au fait que vous avez exécuté la commande `git config` à l'étape 1.7.

```
[student@workstation D0180-apps]$ echo "OCP4" > TEST
[student@workstation D0180-apps]$ git add .
[student@workstation D0180-apps]$ git commit -m "OCP4"
[student@workstation D0180-apps]$ git push
...output omitted...
```

- 5.7. Vérifiez que `/home/student/D0180-apps` contient les exemples d'applications de ce cours et repassez au dossier personnel de l'utilisateur.

```
[student@workstation D0180-apps]$ head README.md
# D0180-apps
...output omitted...

[student@workstation D0180-apps]$ cd ~
[student@workstation ~]$
```

## Fin

Maintenant que vous disposez d'un clone local du référentiel `D0180-apps` sur la machine `workstation` et que vous avez correctement exécuté la commande `lab-configure`, vous êtes prêt à lancer les exercices de ce cours.

Dans le cadre de ce cours, tous les exercices qui génèrent des applications depuis la source commencent à partir de la branche `master` du référentiel Git `D0180-apps`. Les exercices qui apportent des modifications au code source nécessitent que vous créeiez des branches pour héberger vos modifications, de sorte que la branche `master` contienne toujours un bon point de départ connu. Si, pour quelque raison que ce soit, vous devez interrompre ou redémarrer un exercice et que vous devez enregistrer ou abandonner les modifications que vous avez apportées à vos branches Git, reportez-vous à *Annexe D, Commandes Git utiles*.

L'exercice guidé est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Les conteneurs sont des exécutable d'application isolés, créés avec une surcharge minimale.
- Une image de conteneur empaquette une application avec toutes ses dépendances, ce qui simplifie son exécution dans différents environnements.
- Les applications telles que Podman créent des conteneurs à l'aide de fonctions du noyau Linux standard.
- Les registres d'images de conteneur constituent le mécanisme privilégié pour distribuer des images de conteneurs à plusieurs utilisateurs et hôtes.
- OpenShift orchestre les applications composées de plusieurs conteneurs à l'aide de Kubernetes.
- Kubernetes gère l'équilibrage de charge, la haute disponibilité et le stockage persistant pour les applications en conteneur.
- Outre la facilité d'utilisation, OpenShift enrichit Kubernetes de fonctions de multisite, de sécurité et d'intégration continue/déploiement continu.
- Les routes OpenShift permettent un accès externe aux applications en conteneur de manière simple.



## chapitre 2

# Création de services en conteneur

### Objectif

Déployer un serveur en utilisant la technologie des conteneurs.

### Résultats

- Créer un serveur de base de données à partir d'une image de conteneur.

### Sections

- Approvisionnement de services en conteneur
- Crédit d'une instance de base de données MySQL

# Approvisionnement de services en conteneur

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Rechercher et récupérer des images de conteneur avec Podman.
- Exécuter et configurer les conteneurs localement.
- Utiliser Red Hat Container Catalog.

## Récupération d'images de conteneur avec Podman

Les applications peuvent s'exécuter dans des conteneurs, fournissant ainsi un environnement d'exécution isolé et contrôlé. L'exécution d'une application en conteneur, c'est-à-dire l'exécution d'une application à l'intérieur d'un conteneur, nécessite une image de conteneur et un ensemble de systèmes de fichiers qui fournit tous les fichiers de l'application, les bibliothèques et les dépendances nécessaires à l'exécution de l'application. Les images de conteneur se trouvent dans les registres d'images qui permettent aux utilisateurs de rechercher et de récupérer des images de conteneur. Les utilisateurs de Podman peuvent utiliser la sous-commande `search` pour trouver des images disponibles à partir de registres distants ou locaux.

```
[user@demo ~]$ podman search rhel
INDEX      NAME                  DESCRIPTION  STARS OFFICIAL AUTOMATED
redhat.com  registry.access.redhat.com/rhel This plat... 0
...output omitted...
```

Après avoir trouvé une image, vous pouvez utiliser Podman pour la télécharger. Utilisez la sous-commande `pull` pour indiquer à Podman de récupérer l'image et de l'enregistrer localement pour une utilisation ultérieure.

```
[user@demo ~]$ podman pull rhel
Trying to pull registry.access.redhat.com/rhel...
Getting image source signatures
Copying blob sha256: ...output omitted...
  72.25 MB / 72.25 MB [=====] 8s
Copying blob sha256: ...output omitted...
  1.20 KB / 1.20 KB [=====] 0s
Copying config sha256: ...output omitted...
  6.30 KB / 6.30 KB [=====] 0s
Writing manifest to image destination
Storing signatures
699d44bc6ea2b9fb23e7899bd4023d3c83894d3be64b12e65a3fe63e2c70f0ef
```

Les images de conteneur sont nommées en fonction de la syntaxe suivante :

```
registry-name/user-name/image-name:tag
```

Syntaxe de dénomination du Registre :

- **registry-name** est le nom du registre stockant l'image. Il s'agit généralement du nom de domaine complet (FQDN) du registre.
- **user-name** est le nom de l'utilisateur ou de l'organisation auquel l'image appartient.
- **image-name** doit être unique dans l'espace de noms User.
- **tag** identifie la version de l'image. Si le nom de l'image n'inclut aucune balise d'image, la valeur **latest** est utilisée par défaut.



### Note

L'installation Podman de cette salle de classe utilise plusieurs registres disponibles publiquement, tels que Quay.io et Red Hat Container Catalog.

Après la récupération, Podman stocke les images en local et vous pouvez les lister avec la sous-commande **images** :

```
[user@demo ~]$ podman images
REPOSITORY           TAG      IMAGE ID      CREATED       SIZE
registry.access.redhat.com/rhel   latest   699d44bc6ea2  4 days ago  214MB
...output omitted...
```

## Exécution de conteneurs

La commande **podman run** exécute un conteneur localement basé sur une image. Au minimum, la commande nécessite le nom de l'image à exécuter dans le conteneur.

L'image de conteneur spécifie un processus qui commence dans le conteneur et qui est appelé point d'entrée. La commande **podman run** utilise tous les paramètres après le nom de l'image en tant que commande de point d'entrée pour le conteneur. L'exemple suivant démarre un conteneur à partir d'une image Red Hat Universal Base. Il définit le point d'entrée de ce conteneur sur la commande `echo "Hello world"` :

```
[user@demo ~]$ podman run ubi8/ubi:8.3 echo 'Hello world!'
Hello world!
```

Pour démarrer une image de conteneur comme processus d'arrière-plan, transmettez l'option **-d** à la commande **podman run** :

```
[user@demo ~]$ podman run -d -p 8080 registry.redhat.io/rhel8/httpd-24
ff4ec6d74e9b2a7b55c49f138e56f8bc46fe2a09c23093664fea7febc3dfa1b2
[user@demo ~]$ podman port -l
8080/tcp -> 0.0.0.0:44389
[user@demo ~]$ curl http://0.0.0.0:44389
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
...output omitted...
```

Cet exemple exécute un serveur HTTP Apache en conteneur en arrière-plan. Il utilise l'option `-p 8080` pour lier le port du serveur HTTP à un port local. Ensuite, il utilise la commande `podman port` pour récupérer le port local sur lequel le conteneur écoute. Enfin, il utilise ce port pour créer l'URL cible et extraire la page racine du serveur HTTP Apache. Cette réponse prouve que le conteneur est toujours opérationnel après la commande `podman run`.

**Note**

La plupart des sous-commandes de Podman acceptent l'indicateur `-l` (`l` pour `latest`) en remplacement de l'identifiant du conteneur. Cet indicateur applique la commande au dernier conteneur utilisé dans n'importe quelle commande Podman.

**Note**

Si l'image à exécuter n'est pas disponible localement lors de l'utilisation de la commande `podman run`, Podman utilise automatiquement `pull` pour télécharger l'image.

Lorsqu'il est fait référence au conteneur, Podman reconnaît un conteneur avec le nom du conteneur ou l'ID de conteneur généré. Utilisez l'option `--name` pour définir le nom du conteneur lors de son exécution avec Podman. Les noms de conteneurs doivent être uniques. Si la commande `podman run` n'inclut aucun nom de conteneur, Podman génère un nom aléatoire unique.

Si les images nécessitent que l'utilisateur interagissent avec la console, Podman peut alors rediriger les flux d'entrée et de sortie du conteneur vers la console. La sous-commande `run` nécessite les indicateurs `-t` et `-i` (ou l'indicateur `-it`) pour permettre l'inactivité.

**Note**

De nombreux indicateurs Podman ont également une forme longue alternative ; certains d'entre eux sont expliqués ci-dessous :

- `-t` équivaut à `--tty`, ce qui signifie qu'un `pseudo-tty` (pseudo-terminal) doit être alloué au conteneur.
- `-i` est identique à `--interactive`. Lorsqu'elle est utilisée, l'entrée standard est maintenue ouverte dans le conteneur.
- `-d`, ou sa forme longue `--detach`, signifie que le conteneur fonctionne en arrière-plan (détaché). Podman imprime ensuite l'identifiant du conteneur.

Voir la documentation de Podman pour la liste complète des indicateurs.

L'exemple suivant démarre un terminal Bash à l'intérieur du conteneur et y exécute de manière interactive des commandes :

```
[user@demo ~]$ podman run -it ubi8/ubi:8.3 /bin/bash
bash-4.2# ls
...output omitted...
bash-4.2# whoami
root
bash-4.2# exit
exit
[user@demo ~]$
```

Certains conteneurs requièrent ou peuvent utiliser des paramètres externes fournis au démarrage. L'approche la plus courante pour fournir et exploiter ces paramètres consiste à utiliser des variables d'environnement. Podman peut injecter des variables d'environnement dans des conteneurs au démarrage en ajoutant l'indicateur `-e` à la sous-commande `run` :

```
[user@demo ~]$ podman run -e GREET=Hello -e NAME=RedHat \
> ubi8/ubi:8.3 printenv GREET NAME
Hello
RedHat
[user@demo ~]$
```

L'exemple précédent démarre un conteneur d'images UBI qui imprime les deux variables d'environnement fournies en tant que paramètres.

Un autre cas d'utilisation pour les variables d'environnement est la configuration des informations d'identification sur un serveur de base de données MySQL.

```
[user@demo ~]$ podman run --name mysql-custom \
> -e MYSQL_USER=redhat -e MYSQL_PASSWORD=r3dh4t \
> -e MYSQL_ROOT_PASSWORD=r3dh4t \
> -d registry.redhat.io/rhel8/mysql-80
```

## Utilisation de Red Hat Container Catalog

Red Hat conserve son référentiel d'images de conteneur optimisées. L'utilisation de ce référentiel fournit aux clients une couche de protection et de fiabilité par rapport aux vulnérabilités connues, qui pourraient être potentiellement causées par des images non testées. La commande `podman` standard est compatible avec Red Hat Container Catalog. Red Hat Container Catalog fournit une interface conviviale permettant de rechercher et d'explorer les images de conteneur à partir du référentiel Red Hat.

Container Catalog fait également office d'interface unique, fournissant un accès à différents aspects de toutes les images de contenu disponibles dans le référentiel. Cela s'avère utile pour déterminer la meilleure de plusieurs versions d'images de conteneur au moyen de notes d'index de santé. L'index de santé indique l'état actuel d'une image et indique si les dernières mises à jour de sécurité ont été appliquées.

Container Catalog donne également accès à la documentation d'errata pour une image. Il décrit les dernières corrections de bogues et améliorations apportées à chaque mise à jour. Il suggère également la meilleure technique pour extraire une image sur chaque système d'exploitation.

Les images suivantes mettent en évidence quelques-unes des fonctions de Red Hat Container Catalog :

The screenshot shows the Red Hat Container Catalog homepage. At the top, there is a search bar with the query "Apache httpd". Below the search bar, there are filters for "Provider" (IBM, Red Hat, Inc.), "Category" (Database & Data Management, Programming Languages & Runtimes, Web Services), and "Product". The search results are displayed in three cards:

- Red Hat Apache httpd 2.4**: rhel8/httpd-24 by Red Hat, Inc. Platform for running Apache httpd 2.4 or building httpd-based application. Updated 11 hours ago.
- Red Hat Apache httpd 2.4**: rhel8/httpd-24 by Red Hat, Inc. Platform for running Apache httpd 2.4 or building httpd-based application. Updated 11 hours ago.
- IBM Apache CouchDB**: ibm/couchdb3 by IBM Apache CouchDB is a database that uses JSON for documents, an HTTP API, & JavaScript/... Updated 4 days ago.

A "Have feedback?" button is located at the bottom right of the search results area.

**Figure 2.1: Page d'accueil de Red Hat Container Catalog**

Comme illustré dans l'image précédente, la recherche d'Apache `httpd` dans la zone de recherche de Container Catalog affiche une liste de suggestions de produits et de référentiels d'images correspondant au modèle de recherche. Pour accéder à la page d'image Apache `httpd 2.4`, sélectionnez `rhel8/httpd-24` dans la liste suggérée.

Après avoir sélectionné l'image souhaitée, la page suivante fournit des informations supplémentaires sur l'image :

The screenshot shows the details page for the Apache httpd 2.4 image. The top navigation bar includes "Home", "Software", "Container images", and the specific image name "Apache httpd 2.4". The page title is "Apache httpd 2.4". It features a "Standalone Image" section with a "rhel8/httpd-24" card. The card includes fields for "Architecture" (amd64), "Tag" (latest), and a "Provided by" section with the Red Hat logo. Below this is an "Overview" section with tabs for "Overview", "Security", "Packages", "Dockerfile", and "Get this image". The "Overview" tab is selected. The "Description" section contains a detailed description of Apache HTTP Server 2.4. To the right, there are sections for "Published" (28 days ago), "Release category" (Generally Available), "Health index" (B 0 2), and a "Have feedback?" button. At the bottom, there is a note about usage in OpenShift and a link to the Dockerfile.

**Figure 2.2: Page d'image d'aperçu Apache httpd 2.4 (rhel8/httpd-24)**

Le panneau `Apache httpd 2.4` affiche les détails des images et plusieurs onglets. Cette page indique que Red Hat conserve le référentiel d'images.

Sous l'onglet *Overview* figurent d'autres détails :

- *Description* : résumé des capacités de l'image.
- *Documentation* : références à la documentation d'auteur du conteneur.
- *Products using this container* : indique que Red Hat Enterprise Linux utilise ce référentiel d'images.

Sur le côté droit sont affichées des informations sur le moment où l'image a reçu sa dernière mise à jour, la dernière balise appliquée à l'image, son intégrité, sa taille, etc.

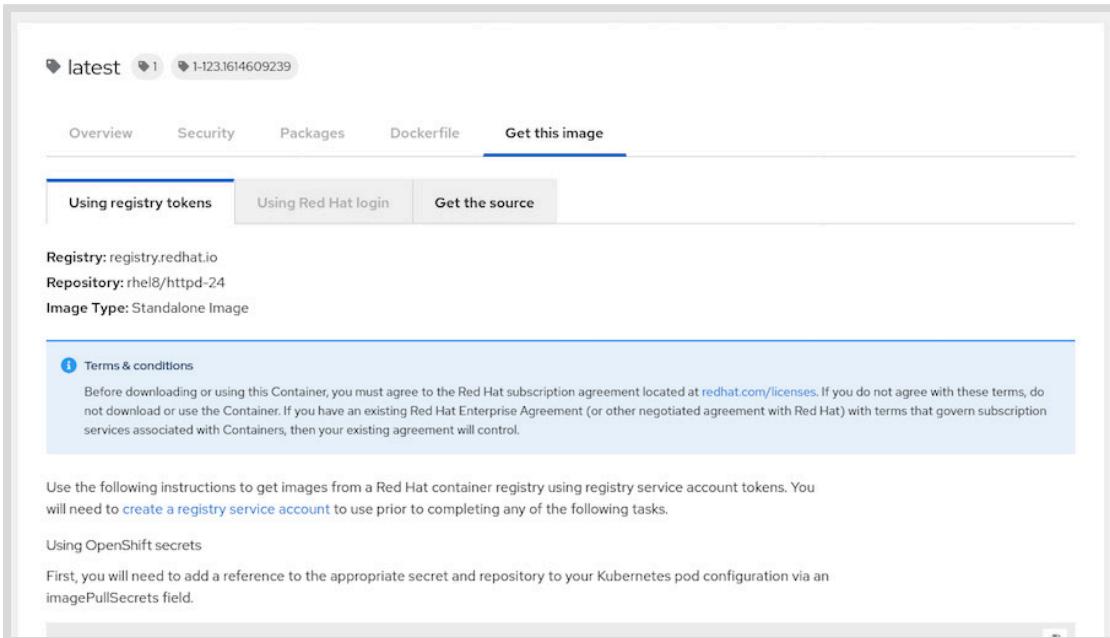


Figure 2.3: Dernière page d'image Apache httpd 2.4 (rhel8/httpd-24)

L'onglet *Get this image* fournit la procédure permettant d'obtenir la version la plus récente de l'image. La page fournit différentes options pour récupérer l'image. Sélectionnez la procédure de votre choix dans les onglets, et la page fournit les instructions appropriées pour récupérer l'image.

 **Références**

**Red Hat Container Catalog**  
<https://registry.redhat.io>

**Site Web Quay.io**  
<https://quay.io>

## ► Exercice guidé

# Création d'une instance de base de données MySQL

Dans cet exercice, vous allez démarrer une base de données MySQL dans un conteneur, puis créer et remplir une base de données.

## Résultats

Vous devriez pouvoir démarrer une base de données à partir d'une image de conteneur et stocker des informations dans la base de données.

## Avant De Commencer

Ouvrez un terminal sur `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab container-create start
```

## Instructions

- ▶ 1. Créez une instance de conteneur MySQL.
  - 1.1. Connectez-vous à Red Hat Container Catalog à l'aide de votre compte Red Hat. Si vous devez vous inscrire auprès de Red Hat, reportez-vous aux instructions dans Annexe C, *Création d'un compte Red Hat*.

```
[student@workstation ~]$ podman login registry.redhat.io
Username: your_username
Password: your_password
Login Succeeded!
```

- 1.2. Démarrez un conteneur à partir de l'image MySQL Red Hat Container Catalog.

```
[student@workstation ~]$ podman run --name mysql-basic \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> -d registry.redhat.io/rhel8/mysql-80:1
Trying to pull ...output omitted...
Copying blob ...output omitted...
Writing manifest to image destination
Storing signatures
2d37682eb33a70330259d6798bdfdc37921367f56b9c2a97339d84faa3446a03
```

Cette commande télécharge l'image de conteneur MySQL 8.0 avec la balise 1, puis démarre un conteneur basé sur celle-ci. Elle crée une base de données intitulée `items`, dont le propriétaire est un utilisateur appelé `user1` avec le mot de passe `mypa55`. Le mot de passe de l'administrateur de base de données est défini sur `r00tpa55` et le conteneur s'exécute en arrière-plan.

- 1.3. Vérifiez que le conteneur a démarré sans erreurs.

```
[student@workstation ~]$ podman ps --format "{{.ID}} {{.Image}} {{.Names}}"
2d37682eb33a registry.redhat.io/rhel8/mysql-80:1 mysql-basic
```

- 2. Accédez au sandbox de conteneur en exécutant la commande suivante :

```
[student@workstation ~]$ podman exec -it mysql-basic /bin/bash
bash-4.4$
```

Cette commande démarre un shell Bash, s'exécutant en tant qu'utilisateur `mysql` dans le conteneur MySQL.

- 3. Ajoutez des données à la base de données.

- 3.1. Connectez-vous à MySQL en tant qu'administrateur de la base de données (root).

Exécutez la commande suivante à partir du terminal de conteneur pour vous connecter à la base de données :

```
bash-4.4$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.
...output omitted...
mysql>
```

La commande `mysql` ouvre l'invite interactive de la base de données MySQL. Exécutez la commande suivante pour déterminer la disponibilité de la base de données :

```
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| items          |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.01 sec)
```

- 3.2. Créez une nouvelle table dans la base de données `items`. Exécutez la commande suivante pour accéder à la base de données.

```
mysql> USE items;
Database changed
```

- 3.3. Créez une table appelée `Projects` dans la base de données `items`.

**chapitre 2 |** Création de services en conteneur

```
mysql> CREATE TABLE Projects (id int NOT NULL,  
-> name varchar(255) DEFAULT NULL,  
-> code varchar(255) DEFAULT NULL,  
-> PRIMARY KEY (id));  
Query OK, 0 rows affected (0.01 sec)
```

Vous pouvez éventuellement utiliser le fichier ~/D0180/solutions/container-create/create\_table.txt pour copier-coller l'instruction MySQL CREATE TABLE fournie.

3.4. Utilisez la commande `show tables` pour vérifier que la table a été créée.

```
mysql> SHOW TABLES;  
+-----+  
| Tables_in_items |  
+-----+  
| Projects |  
+-----+  
1 row in set (0.00 sec)
```

3.5. Utilisez la commande `insert` pour insérer une ligne dans la table.

```
mysql> INSERT INTO Projects (id, name, code) VALUES (1, 'DevOps', 'D0180');  
Query OK, 1 row affected (0.02 sec)
```

3.6. Utilisez la commande `select` pour vérifier que les informations du projet ont été ajoutées à la table.

```
mysql> SELECT * FROM Projects;  
+---+-----+---+  
| id | name | code |  
+---+-----+---+  
| 1 | DevOps | D0180 |  
+---+-----+  
1 row in set (0.00 sec)
```

3.7. Quittez l'invite MySQL et le conteneur MySQL.

```
mysql> exit  
Bye  
bash-4.4$ exit  
exit
```

## Fin

Sur workstation, exéutez le script `lab container-create finish` pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab container-create finish
```

Cela met fin à cet exercice.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Podman permet aux utilisateurs de rechercher et de télécharger des images à partir de registres locaux ou distants.
- La commande `podman run` crée et démarre un conteneur à partir d'une image conteneur.
- Les conteneurs sont exécutés en arrière-plan à l'aide de l'indicateur `-d`, ou de manière interactive en utilisant l'indicateur `-it`.
- Certaines images de conteneur peuvent nécessiter des variables d'environnement définies à l'aide de l'option `-e` à partir de la commande `podman run`.
- Red Hat Container Catalog facilite la recherche, l'exploration et l'analyse d'images de conteneur dans le référentiel d'images de conteneur officiel de Red Hat.



## chapitre 3

# Gestion des conteneurs

### Objectif

Utiliser des images de conteneur préétablies pour créer et gérer des services en conteneur.

### Résultats

- Gérer le cycle de vie d'un conteneur, de la création à la suppression.
- Enregistrer les données d'application de conteneur avec le stockage persistant.
- Décrire la façon d'utiliser la redirection de port pour accéder à un conteneur.

### Sections

- Gestion du cycle de vie des conteneurs (et exercice guidé)
- Association du stockage persistant à des conteneurs (avec exercice guidé)
- Accès aux conteneurs (avec exercice guidé)

### Atelier

- Gestion des conteneurs

# Gestion du cycle de vie des conteneurs

## Résultats

À la fin de cette section, les stagiaires seront en mesure de gérer le cycle de vie d'un conteneur de la création à la suppression.

## Gestion du cycle de vie des conteneurs avec Podman

Dans les chapitres précédents, vous avez appris à utiliser Podman pour créer un service en conteneur. Vous allez maintenant approfondir les commandes et les stratégies que vous pouvez utiliser pour gérer le cycle de vie d'un conteneur. Podman vous permet non seulement d'exécuter des conteneurs, mais également de les exécuter en arrière-plan, d'exécuter de nouveaux processus à l'intérieur de ceux-ci et de leur fournir des ressources, telles que des volumes de système de fichiers ou un réseau.

Podman, implémenté par la commande `podman`, fournit un ensemble de sous-commandes permettant de créer et gérer des conteneurs. Les développeurs utilisent ces sous-commandes pour gérer le cycle de vie des conteneurs et des images de conteneur. La figure suivante affiche un résumé des sous-commandes les plus couramment utilisées qui modifient l'état du conteneur et de l'image :

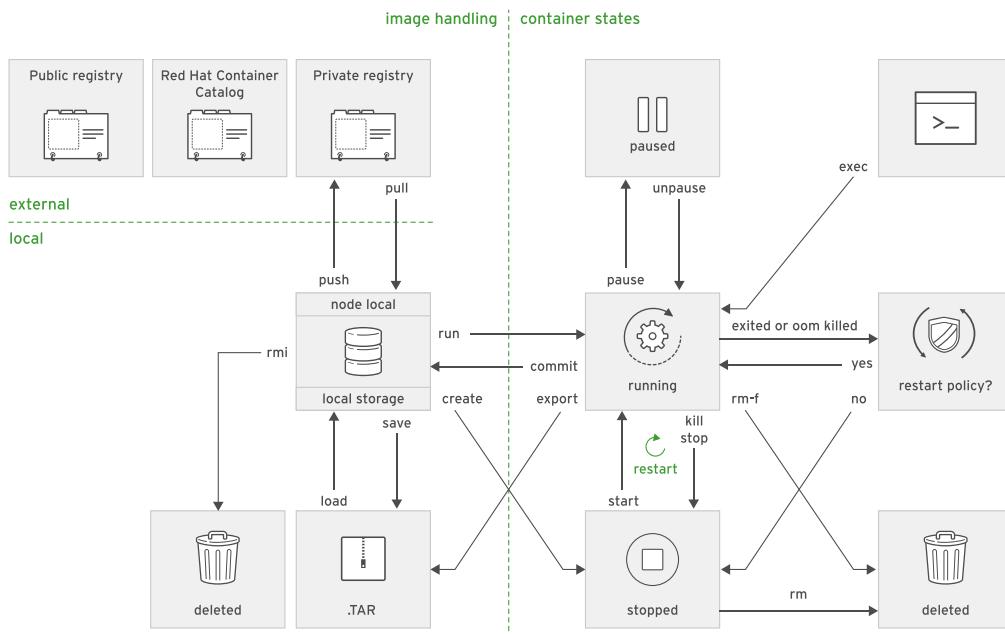


Figure 3.1: Gestion des sous-commandes par Podman

Podman fournit également un ensemble de sous-commandes utiles pour obtenir des informations sur les conteneurs en cours d'exécution et arrêtés.

Vous pouvez utiliser ces sous-commandes pour extraire des informations des conteneurs et des images à des fins de débogage, de mise à jour ou de création de rapports. La figure suivante

présente un récapitulatif des sous-commandes les plus couramment utilisées pour effectuer des requêtes à partir de conteneurs et d'images :

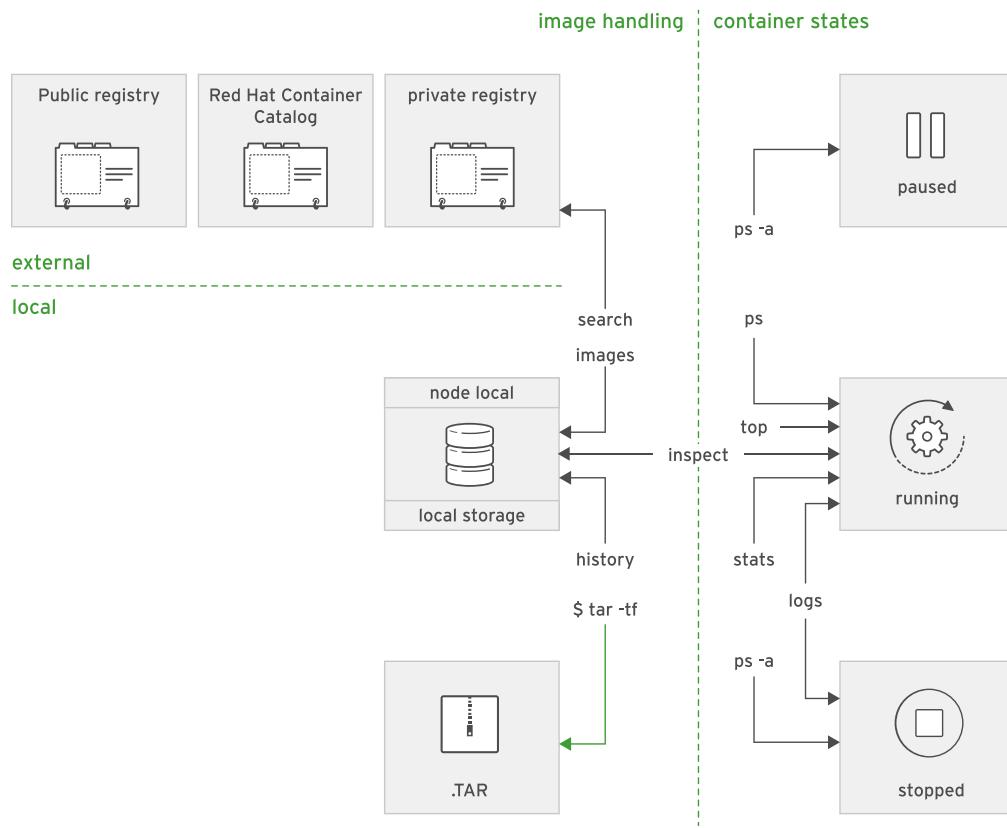


Figure 3.2: Sous-commandes de requête Podman

Référez-vous à ces deux illustrations pour vous aider dans votre apprentissage des sous-commandes Podman tout au long de ce cours.

## Création de conteneurs

La commande `podman run` crée un conteneur à partir d'une image et démarre un processus à l'intérieur du nouveau conteneur. Si l'image de conteneur n'est pas disponible localement, cette commande tente de télécharger l'image à l'aide du référentiel d'images configuré :

```
[user@host ~]$ podman run registry.redhat.io/rhel8/httpd-24
Trying to pull registry.redhat.io/rhel8/httpd-24...
Getting image source signatures
Copying blob sha256:23113...b0be82
72.21 MB / 72.21 MB [=====] 7s
...output omitted... AH00094: Command line: 'httpd -D FOREGROUND'
^C
```

Dans cet exemple de sortie, le conteneur a été démarré avec un processus non interactif (sans l'option `-it`) et s'exécute au premier plan car il n'a pas été démarré avec l'option `-d`. Arrêter le processus résultant avec `Ctrl+C` (`SIGINT`) par conséquent, arrête le processus de conteneur ainsi que le conteneur lui-même.

### chapitre 3 | Gestion des conteneurs

Podman identifie les conteneurs par un ID ou un nom de conteneur unique. La commande `podman ps` affiche l'ID de conteneur et les noms de tous les conteneurs en cours d'exécution :

```
[user@host ~]$ podman ps
CONTAINER ID IMAGE COMMAND ... NAMES
47c9aad6049①
registry.redhat.io/rhel8/httpd-24 "/usr/bin/run-http..." ... focused_fermat②
```

- ① L'ID de conteneur est unique et généré automatiquement.
- ② Le nom du conteneur peut être spécifié manuellement, sinon il est généré automatiquement. Ce nom doit être unique, sinon la commande `run` échoue.

La commande `podman run` génère automatiquement un identifiant unique et aléatoire. Elle génère également un nom de conteneur aléatoire. Pour définir explicitement le nom du conteneur, utilisez l'option `--name` lors de l'exécution d'un conteneur :

```
[user@host ~]$ podman run --name my-httpd-container \
> registry.redhat.io/rhel8/httpd-24
...output omitted... AH00094: Command line: 'httpd -D FOREGROUND'
```



#### Note

Le nom doit être unique. Podman renvoie une erreur si le nom est déjà utilisé par un conteneur quelconque, y compris les conteneurs arrêtés.

Une autre fonction importante est la possibilité d'exécuter le conteneur en tant que processus démon en arrière-plan. L'option `-d` sert à l'exécution en mode détaché. Lorsque vous utilisez cette option, Podman renvoie l'ID de conteneur à l'écran, vous permettant de continuer à exécuter des commandes dans le même terminal pendant que le conteneur s'exécute en arrière-plan :

```
[user@host ~]$ podman run --name my-httpd-container \
> -d registry.redhat.io/rhel8/httpd-24
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

L'image de conteneur spécifie la commande à exécuter pour démarrer le processus en conteneur, appelé point d'entrée. La commande `podman run` peut remplacer ce point d'entrée en incluant la commande après l'image du conteneur :

```
[user@host ~]$ podman run registry.redhat.io/rhel8/httpd-24 ls /tmp
ks-script-1j4CXN
```

La commande spécifiée doit être exécutable à l'intérieur de l'image de conteneur.



#### Note

Étant donné qu'une commande spécifiée apparaît dans l'exemple, le conteneur ignore le point d'entrée de l'image `httpd`. Par conséquent, le service `httpd` ne démarre pas.

### chapitre 3 | Gestion des conteneurs

Certains conteneurs doivent être exécutés en tant que shell ou processus interactif. Cela inclut les conteneurs exécutant des processus nécessitant une intervention de l'utilisateur (comme la saisie de commandes), et les processus qui génèrent une sortie via une sortie standard. L'exemple suivant démarre un shell bash interactif dans un conteneur `registry.redhat.io/rhel8/httpd-24` :

```
[user@host ~]$ podman run -it registry.redhat.io/rhel8/httpd-24 /bin/bash  
bash-4.4#
```

- Les options `-t` et `-i` permettent la redirection de terminal pour les programmes textuels interactifs.
- L'option `-t` alloue un `pseudo-tty` (un terminal) et l'attache à l'entrée standard du conteneur.
- L'option `-i` permet de garder l'entrée standard du conteneur ouverte, même s'il a été détaché, afin que le processus principal puisse continuer à attendre l'entrée.

## Exécution de commandes dans un conteneur

Lorsqu'un conteneur démarre, il exécute la commande de point d'entrée. Toutefois, il peut être nécessaire d'exécuter d'autres commandes pour gérer le conteneur en cours d'exécution.

Voici des cas d'utilisation typiques :

- Exécution d'un shell interactif dans un conteneur déjà en cours d'exécution.
- Processus en cours qui mettent à jour ou affichent les fichiers du conteneur.
- Démarrage de nouveaux processus en arrière-plan dans le conteneur.

La commande `podman exec` démarre un processus supplémentaire dans un conteneur déjà en cours d'exécution :

```
[user@host ~]$ podman exec 7ed6e671a600 cat /etc/hostname  
7ed6e671a600
```

Dans cet exemple, l'ID de conteneur est utilisé pour exécuter une commande dans un conteneur existant.

Podman se souvient du dernier conteneur créé. Les développeurs peuvent ignorer l'écriture de l'ID ou du nom de ce conteneur dans les commandes Podman ultérieures en remplaçant l'ID du conteneur par l'option `-l` (ou `--latest`) :

```
[user@host ~]$ podman exec my-httpd-container cat /etc/hostname  
7ed6e671a600  
  
[user@host ~]$ podman exec -l cat /etc/hostname  
7ed6e671a600
```

## Gestion des conteneurs

La création et le démarrage d'un conteneur ne constituent que la première étape du cycle de vie du conteneur. Ce cycle de vie inclut également l'arrêt, le redémarrage ou la suppression du conteneur. Les utilisateurs peuvent également examiner l'état du conteneur et les métadonnées à des fins de débogage, de mise à jour ou de création de rapports.

### chapitre 3 | Gestion des conteneurs

Podman fournit les commandes suivantes pour gérer les conteneurs :

`podman ps`

Cette commande liste les conteneurs en cours d'exécution :

```
[user@host ~]$ podman ps
CONTAINER ID  IMAGE          COMMAND      CREATED     STATUS      PORTS      NAMES
77d4b7b8ed1f  registry.redhat.io/rhel8/httpd-24  /usr/bin/run-httpd...  ...ago
Up...  my-htt... ❶
```

- ❶ Chaque ligne décrit les informations relatives au conteneur.

#### CONTAINER ID

Une fois créé, chaque conteneur obtient un **container ID** qui est un nombre hexadécimal. Cet ID ressemble à un ID d'image mais n'a pas de relation.

#### IMAGE

Le champ **IMAGE** indique l'image de conteneur utilisée pour démarrer le conteneur.

#### COMMAND

Le champ **COMMAND** indique la commande exécutée au démarrage du conteneur.

#### CREATED

Le champ **CREATED** indique la date et l'heure de démarrage du conteneur.

#### STATUS

Le champ **STATUS** indique la durée totale de fonctionnement du conteneur, s'il est encore en cours d'exécution, ou le temps passé depuis son arrêt.

#### PORTS

Le champ **PORTS** indique les ports exposés par le conteneur ou tout transfert de port pouvant être configuré.

#### NAMES

Le champ **NAMES** indique le nom du conteneur.

Podman ne supprime pas immédiatement les conteneurs arrêtés. Podman préserve ses systèmes de fichiers locaux et d'autres états pour faciliter une analyse *post-mortem*. Option **-a** lists all containers, including stopped ones:

```
[user@host ~]$ podman ps -a
CONTAINER ID  IMAGE          COMMAND      CREATED     STATUS      PORTS      NAMES
4829d82fbbff  registry.redhat.io/rhel8/httpd-24  /usr/bin/run-httpd...  ...ago
Exited (0)...  my-htt... ❷
```



#### Note

Lors de la création de conteneurs, Podman abandonne si le nom du conteneur est déjà utilisé, même s'il a le statut stopped. Cette option aide à éviter les noms de conteneurs en double.

`podman stop`

Cette commande arrête correctement un conteneur en cours d'exécution :

```
[user@host ~]$ podman stop my-httdp-container  
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

La commande `podman stop` est le moyen le plus facile pour identifier le processus de démarrage du conteneur sur le système d'exploitation hôte et l'arrêter.

#### podman kill

Cette commande envoie des signaux Unix au processus principal du conteneur. Si aucun signal n'est spécifié, elle envoie le message `SIGKILL`, en terminant le processus principal et le conteneur.

```
[user@host ~]$ podman kill my-httdp-container  
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

Vous pouvez spécifier le signal avec l'option `-s` :

```
[user@host ~]$ podman kill -s SIGKILL my-httdp-container  
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

Tout signal Unix peut être envoyé au processus principal. Podman accepte le numéro ou le nom du signal.

Le tableau suivant montre plusieurs signaux utiles :

Signal	Valeur	Action par défaut	Commentaire
SIGHUP	1	Term	Hangup détecté sur le terminal de contrôle ou mort du processus de contrôle
SIGINT	2	Term	Interruption du clavier
SIGQUIT	3	Core	Quitter à partir du clavier
SIGILL	4	Core	Instruction illégale
SIGABRT	6	Core	Signal d'abandon de <code>abort(3)</code>
SIGFPE	8	Core	Exception en virgule flottante
SIGKILL	9	Term	Supprimer le signal
SIGSEGV	11	Core	Référence mémoire invalide
SIGPIPE	13	Term	Pipe cassé : écrivez vers le pipe sans lecteur
SIGALRM	14	Term	Signal de minuteur de <code>alarm(2)</code>
SIGTERM	15	Term	Signal d'arrêt
SIGUSR1	30,10,16	Term	Signal 1 défini par l'utilisateur
SIGUSR2	31,12,17	Term	Signal 2 défini par l'utilisateur
SIGCHLD	20,17,18	Ign	Enfant arrêté ou terminé

Signal	Valeur	Action par défaut	Commentaire
SIGCONT	19,18,25	Cont	Continuer si arrêté
SIGSTOP	17,19,23	Stop	Arrêter le processus
SIGTSTP	18,20,24	Stop	Arrêt de la saisie sur le tty
SIGTTIN	21,21,26	Stop	Saisie tty pour le processus d'arrière-plan
SIGTTOUT	22,22,27	Stop	Sortie tty pour le processus d'arrière-plan

L'action par défaut pour chaque signal est spécifiée comme suit :

Action par défaut	Description
Term	Arrêter le processus.
Core	Terminer le processus et générer un vidage mémoire.
Ign	Le signal est ignoré.
Stop	Arrêter le processus.

Voici d'autres commandes podman utiles :

#### podman restart

Cette commande redémarre un conteneur arrêté :

```
[user@host ~]$ podman restart my-httdp-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

La commande `podman restart` crée un conteneur avec le même ID de conteneur en réutilisant l'état du conteneur arrêté et le système de fichiers.

#### podman rm

Cette commande supprime un conteneur, et ignore son état et son système de fichiers.

```
[user@host ~]$ podman rm my-httdp-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

L'option `-f` de la sous-commande `rm` indique à Podman de supprimer le conteneur même s'il n'est pas arrêté. Cette option termine le conteneur de force, puis le supprime. L'utilisation de l'option `-f` revient à utiliser les commandes `podman kill` et `podman rm` ensemble.

Vous pouvez supprimer simultanément tous les conteneurs. Beaucoup de sous-commandes `podman` acceptent l'option `-a`. Cette option indique que la sous-commande est utilisée sur tous les conteneurs ou images disponibles. L'exemple suivant supprime tous les conteneurs :

```
[user@host ~]$ podman rm -a
5fd8e98ec7eab567eabe84943fe82e99fdfc91d12c65d99ec760d5a55b8470d6
716fd687f65b0957edac73b84b3253760e915166d3bc620c4aec8e5f4eadfe8e
86162c906b44f4cb63ba2e3386554030dcbb6abedbce9e9fcad60aa9f8b2d5d4
```

Avant de supprimer tous les conteneurs, tous les conteneurs en cours d'exécution doivent avoir le statut stopped. Vous pouvez utiliser la commande suivante pour arrêter tous les conteneurs :

```
[user@host ~]$ podman stop -a  
5fd8e98ec7eab567eabe84943fe82e99fdfc91d12c65d99ec760d5a55b8470d6  
716fd687f65b0957edac73b84b3253760e915166d3bc620c4aec8e5f4eadfe8e  
86162c906b44f4cb63ba2e3386554030dcba6abedbce9e9fcad60aa9f8b2d5d4
```



### Note

Les sous-commandes `inspect`, `stop`, `kill`, `restart` et `rm` peuvent utiliser l'ID de conteneur à la place du nom de conteneur.



### Références

#### Page de manuel Unix Posix Signals

<http://man7.org/linux/man-pages/man7/signal.7.html>

## ► Exercice guidé

# Gestion d'un conteneur MySQL

Dans cet exercice, vous allez créer et gérer un conteneur de base de données MySQL®.

## Résultats

Vous devez pouvoir créer et gérer un conteneur de base de données MySQL.

## Avant De Commencer

Assurez-vous que la commande `workstation` est disponible sur la machine `podman` et qu'elle est correctement configurée en exécutant la commande suivante à partir d'une fenêtre de terminal :

```
[student@workstation ~]$ lab manage-lifecycle start
```

## Instructions

- 1. Téléchargez l'image de conteneur de base de données MySQL et essayez de la démarrer. Le conteneur ne démarre pas, car plusieurs variables d'environnement doivent être fournies à l'image.
- 1.1. Connectez-vous à Red Hat Container Catalog à l'aide de votre compte Red Hat. Si vous devez vous inscrire auprès de Red Hat, reportez-vous aux instructions dans *Annexe C, Création d'un compte Red Hat*.

```
[student@workstation ~]$ podman login registry.redhat.io
Username: your-username
Password: your-password
Login Succeeded!
```

- 1.2. Téléchargez l'image de conteneur de base de données MySQL et essayez de la démarrer.

```
[student@workstation ~]$ podman run --name mysql-db \
> registry.redhat.io/rhel8/mysql-80:1
Trying to pull ...output omitted...
...output omitted...
Writing manifest to image destination
Storing signatures
You must either specify the following environment variables:
  MYSQL_USER      (regex: ...)
  MYSQL_PASSWORD  (regex: ...)
  MYSQL_DATABASE  (regex: ...)
Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: ...)
```

**Or both.**

Optional Settings:  
*...output omitted...*

**Note**

Si vous essayez d'exécuter le conteneur en tant que démon (-d), alors le message d'erreur mentionnant les variables requises ne s'affiche pas. Cependant, ce message est inclus dans les journaux du conteneur. Vous pouvez afficher ces derniers à l'aide de la commande suivante :

```
[student@workstation ~]$ podman logs mysql-db
```

- ▶ 2. Créez un conteneur appelé `mysql`, puis spécifiez chaque variable requise à l'aide du paramètre `-e`.

**Note**

Prenez soin de démarrer le nouveau conteneur avec le nom correct.

```
[student@workstation ~]$ podman run --name mysql \
> -d -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
```

La commande affiche l'ID de conteneur pour le conteneur `mysql`. Vous trouverez ci-dessous un exemple de sortie.

```
a8a6090a0a632b5876001725b581fdd331c9ab8b9eda21cc2a2899c23f078509
```

- ▶ 3. Vérifiez que le conteneur `mysql` a été correctement démarré. Exécutez la commande suivante :

```
[student@workstation ~]$ podman ps
CONTAINER ID  ... STATUS          ... NAMES
a8a6090a0a63  ... Up 13 seconds ago ... mysql
```

La commande affiche uniquement les 12 premiers caractères de l'ID de conteneur affiché dans la commande précédente.

- ▶ 4. Renseignez la base de données `items` avec la table `Projects` :

- 4.1. Exécutez la commande `podman cp` pour copier le fichier de base de données dans le conteneur `mysql`.

```
[student@workstation ~]$ podman cp ~/DO180/labs/manage-lifecycle/db.sql mysql:/
```

- 4.2. Renseignez la base de données `items` avec la table `Projects`.

```
[student@workstation ~]$ podman exec mysql /bin/bash -c \  
> 'mysql -uuser1 -pmypa55 items < /db.sql'  
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- ▶ 5. Créez un autre conteneur à l'aide de la même image de conteneur que le conteneur précédent. Entrez de manière interactive le shell /bin/bash au lieu d'utiliser la commande par défaut pour l'image de conteneur.

```
[student@workstation ~]$ podman run --name mysql-2 -it \  
> registry.redhat.io/rhel8/mysql-80:1 /bin/bash  
bash-4.4$
```

- ▶ 6. Essayez de vous connecter à la base de données MySQL dans le nouveau conteneur :

```
bash-4.4$ mysql -uroot
```

L'erreur suivante s'affiche :

```
ERROR 2002 (HY000): Can't connect to local MySQL ...output omitted...
```

Le serveur de base de données MySQL n'est pas en cours d'exécution car le conteneur a exécuté la commande /bin/bash au lieu de démarrer le serveur MySQL.

- ▶ 7. Quittez le conteneur.

```
bash-4.4$ exit
```

- ▶ 8. Vérifiez que le conteneur mysql-2 n'est pas en cours d'exécution.

```
[student@workstation ~]$ podman ps -a  
CONTAINER ID  ... STATUS  ... NAMES  
27b4b00b7f5c  ... Exited (1) 4 minutes ago  ... mysql-db  
a8a6090a0a63  ... Up 4 minutes ago  ... mysql  
bd517765c217  ... Exited (0) 8 seconds ago  ... mysql-2
```

- ▶ 9. Interrogez le conteneur mysql pour afficher la liste de toutes les lignes de la table Projects. La commande ordonne au shell bash d'interroger la base de données items à l'aide d'une commande mysql.

```
[student@workstation ~]$ podman exec mysql /bin/bash -c \  
> 'mysql -uuser1 -pmypa55 -e "select * from items.Projects;"'  
mysql: [Warning] Using a password on the command-line interface can be insecure.  
id      name      code  
1       DevOps    D0180
```

## Fin

Sur workstation, exécutez le script suivant pour mettre fin à cet exercice.

```
[student@workstation ~]$ lab manage-lifecycle finish
```

Cela met fin à cet exercice.

# Association du stockage persistant à des conteneurs

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Enregistrer les données d'application lors des suppressions de conteneur grâce à l'utilisation du stockage persistant.
- Configurer les répertoires hôtes à utiliser en tant que volumes de conteneur.
- Monter un volume à l'intérieur du conteneur.

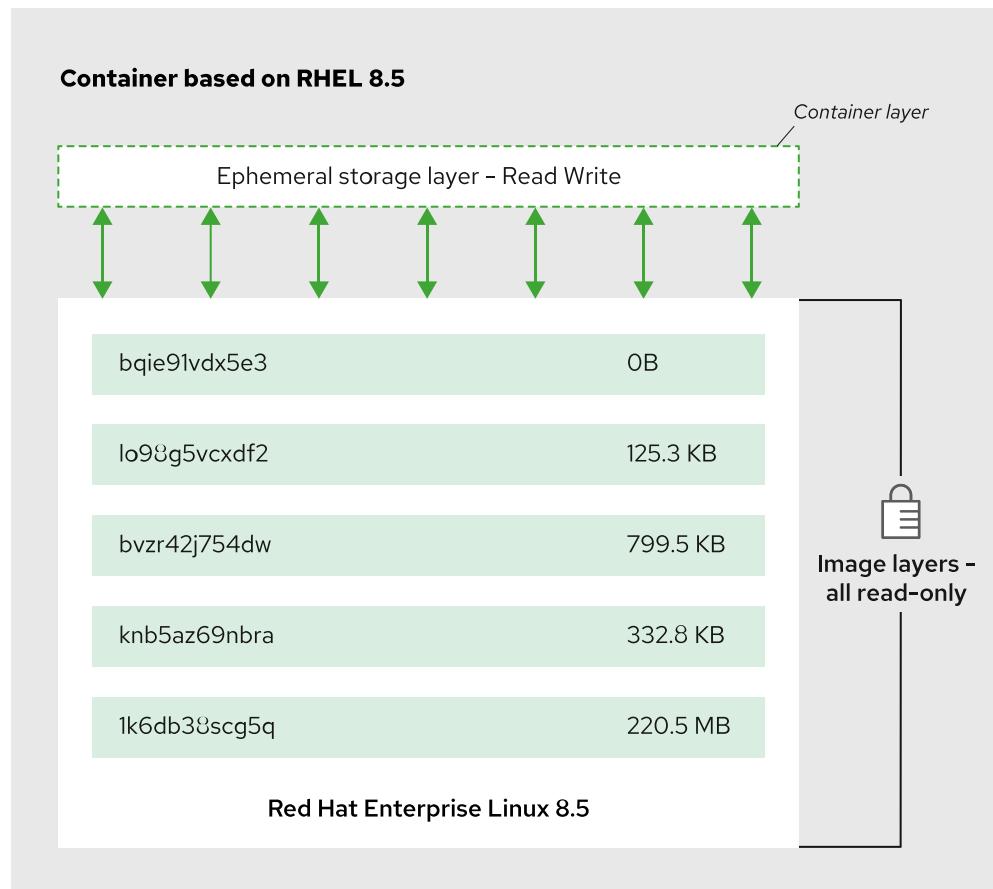
## Préparation d'emplacements de stockage permanent

Le stockage de conteneur est réputé éphémère, ce qui signifie que son contenu n'est pas conservé après la suppression du conteneur. Les applications en conteneur fonctionnent selon l'hypothèse qu'elles commencent toujours avec un stockage vide. La création et la destruction de conteneurs sont, pour cette raison, des opérations relativement économiques.

Précédemment dans ce cours, les images de conteneur ont été définies comme *immuables* et *en plusieurs couches*, ce qui signifie qu'elles ne sont jamais modifiées mais plutôt composées de couches qui permettent d'ajouter ou de supprimer le contenu de couches inférieures.

Un conteneur en cours d'exécution obtient une nouvelle couche sur son image de conteneur de base, et cette couche est le *stockage de conteneur*. Au début, cette couche n'est que le stockage en lecture-écriture du conteneur, et il ne sert qu'à créer des fichiers de travail, des fichiers temporaires et des fichiers journaux. Ces fichiers sont considérés comme étant volatiles. Une application ne cesse pas de fonctionner si ces derniers sont perdus. La couche de stockage de conteneur est exclusive au conteneur en cours d'exécution. Par conséquent, si un autre conteneur est créé à partir de la même image de base, il obtient une autre couche de lecture-écriture. Cela garantit que les ressources de chaque conteneur sont isolées des autres conteneurs similaires.

Le stockage de conteneur éphémère n'est *pas* suffisant pour les applications qui doivent conserver les données au-delà de la durée de vie du conteneur, comme les bases de données. Pour prendre en charge les applications de ce type, l'administrateur doit fournir un conteneur avec du stockage persistant.

**Figure 3.3: Couches du conteneur**

Les applications en conteneur ne doivent pas essayer d'utiliser le stockage de conteneur pour stocker des données persistantes, car elles ne peuvent pas contrôler la durée de conservation du contenu.

Même s'il était possible de conserver indéfiniment le stockage du conteneur, le système de fichiers en couches ne fonctionne pas correctement dans le cas de charges de travail d'E/S intensives et s'avère inadapté pour la plupart des applications nécessitant du stockage persistant.

## Récupération de stockage

Podman conserve l'ancien stockage de conteneur arrêté afin qu'il soit accessible dans le cas d'opérations de résolution de problèmes, par exemple l'examen des messages d'erreur des journaux de conteneurs défaillants.

Si l'administrateur doit récupérer l'ancien stockage de conteneur, alors vous pouvez supprimer le conteneur à l'aide de `podman rm container_id`. Cette commande supprime également le stockage de conteneur. Vous pouvez trouver les ID de conteneur arrêtés à l'aide de la commande `podman ps -a`.

## Préparation du répertoire hôte

Podman peut monter des répertoires d'hôtes dans un conteneur en cours d'exécution. L'application en conteneur considère ces répertoires hôtes comme faisant partie du stockage de conteneurs, un peu comme les applications standard voient un volume réseau distant comme s'il faisait partie du système de fichiers hôte. Toutefois, vous ne pouvez pas récupérer le contenu

### chapitre 3 | Gestion des conteneurs

des répertoires d'hôtes après l'arrêt du conteneur, de sorte que vous pouvez le monter sur de nouveaux conteneurs si nécessaire.

Par exemple, un conteneur de base de données peut utiliser un répertoire hôte pour stocker des fichiers de base de données. Si ce conteneur de base de données échoue, Podman peut créer un nouveau conteneur en utilisant le même répertoire hôte et conserver les données de la base de données à la disposition des applications clientes. En ce qui concerne le conteneur de base de données, l'emplacement de stockage du répertoire hôte n'est pas important du point de vue de l'hôte ; cela peut être une partition locale de disque dur ou un système de fichiers en réseau distant.

Un conteneur est exécuté en tant que processus de système d'exploitation hôte, sous un ID de groupe et d'utilisateur du système d'exploitation hôte. C'est pourquoi vous devez configurer le répertoire hôte avec la propriété et les autorisations permettant l'accès au conteneur. Dans RHEL, vous devez également configurer le répertoire hôte avec le contexte SELinux approprié, à savoir `container_file_t`. Podman utilise le contexte SELinux `container_file_t` pour limiter les fichiers sur le système hôte auxquels le conteneur est autorisé à accéder. Cela évite les fuites d'informations entre le système hôte et les applications s'exécutant à l'intérieur des conteneurs.

Vous pouvez configurer le répertoire hôte comme suit :

1. Créez un répertoire :

```
[user@host ~]$ mkdir /home/student/dbfiles
```

2. L'utilisateur exécutant des processus dans le conteneur doit être capable d'écrire des fichiers dans le répertoire. Vous devez définir l'autorisation avec l'ID numérique d'utilisateur (UID) du conteneur. Dans le cas du service MySQL fourni par Red Hat, l'UID est 27. La commande `podman unshare` fournit une session pour exécuter les commandes au sein du même espace de noms utilisateur que le processus s'exécutant à l'intérieur du conteneur.

```
[user@host ~]$ podman unshare chown -R 27:27 /home/student/dbfiles
```

3. Appliquez le contexte `container_file_t` au répertoire (et tous les sous-répertoires) pour permettre aux conteneurs d'accéder à l'ensemble de son contenu.

```
[user@host ~]$ sudo semanage fcontext -a -t container_file_t \
  '/home/student/dbfiles(/.*)?'
```

4. Appliquez la politique de conteneur SELinux que vous avez configurée lors de la première étape au nouveau répertoire créé :

```
[user@host ~]$ sudo restorecon -Rv /home/student/dbfiles
```

Le répertoire hôte doit être configuré avant de démarrer le conteneur qui l'utilise.

## Montage de volume

Après avoir créé et configuré le répertoire hôte, l'étape suivante consiste à monter ce répertoire dans un conteneur. Pour monter par liaison un répertoire hôte dans un conteneur, ajoutez l'option `-v` à la commande `podman run` en spécifiant le chemin d'accès au répertoire hôte et le chemin d'accès au stockage du conteneur, séparés par le signe deux-points (`:`).

Par exemple, afin d'utiliser le répertoire hôte `/home/student/dbfiles` pour les fichiers de base de données du serveur MySQL, censés se trouver sous `/var/lib/mysql` à l'intérieur d'une image de conteneur MySQL nommée `mysql`, utilisez la commande suivante :

```
[user@host ~]$ podman run -v /home/student/dbfiles:/var/lib/mysql rhmap47/mysql
```

Dans cette commande, si `/var/lib/mysql` existe déjà à l'intérieur d'une image de conteneur `mysql`, le montage de `/home/student/dbfiles` couvre mais ne supprime pas le contenu de l'image de conteneur. Si le montage est supprimé, le contenu d'origine est à nouveau accessible.



## Références

**Gestion des espaces de noms des utilisateurs et SELinux des conteneurs sans racine**

<https://www.redhat.com/sysadmin/user-namespaces-selinux-rootless-containers>

**Présentation des étiquettes SELinux pour les exécutables de conteneur**

<https://opensource.com/article/18/2/selinux-labels-container-runtimes>

## ► Exercice guidé

# Créer un conteneur MySQL avec une base de données persistante

Dans cet exercice, vous allez créer un conteneur qui permet de stocker les données de la base de données MySQL dans un répertoire hôte.

## Résultats

Vous devez pouvoir déployer un conteneur avec une base de données persistante.

## Avant De Commencer

Aucune image de conteneur ne doit être en cours d'exécution sur workstation.

Exécutez la commande suivante sur workstation :

```
[student@workstation ~]$ lab manage-storage start
```

## Instructions

- 1. Créez le répertoire /home/student/local/mysql avec les permissions et le contexte SELinux corrects.

- 1.1. Créez le répertoire /home/student/local/mysql.

```
[student@workstation ~]$ mkdir -vp /home/student/local/mysql
mkdir: created directory /home/student/local
mkdir: created directory /home/student/local/mysql
```

- 1.2. Ajoutez le contexte SELinux approprié pour le répertoire /home/student/local/mysql et son contenu.

```
[student@workstation ~]$ sudo semanage fcontext -a \
> -t container_file_t '/home/student/local/mysql(/.*)?'
```

- 1.3. Appliquez la politique SELinux au répertoire nouvellement créé.

```
[student@workstation ~]$ sudo restorecon -R /home/student/local/mysql
```

- 1.4. Vérifiez que le type de contexte SELinux pour le répertoire /home/student/local/mysql est container\_file\_t.

```
[student@workstation ~]$ ls -ldZ /home/student/local/mysql
drwxrwxr-x. 2 student student unconfined_u:object_r:container_file_t:s0 6 May 26
14:33 /home/student/local/mysql
```

**chapitre 3 |** Gestion des conteneurs

15. Remplacez le propriétaire du répertoire /home/student/local/mysql par l'utilisateur mysql et le groupe mysql :

```
[student@workstation ~]$ podman unshare chown 27:27 /home/student/local/mysql
```

**Note**

L'utilisateur exécutant des processus dans le conteneur doit être capable d'écrire des fichiers dans le répertoire.

Vous devez définir l'autorisation avec l'ID numérique d'utilisateur (UID) du conteneur. Dans le cas du service MySQL fourni par Red Hat, l'UID est 27. La commande `podman unshare` fournit une session pour exécuter les commandes au sein du même espace de noms utilisateur que le processus s'exécutant à l'intérieur du conteneur.

- ▶ 2. Créez une instance de conteneur MySQL avec du stockage persistant.

- 2.1. Connectez-vous à Red Hat Container Catalog à l'aide de votre compte Red Hat. Si vous devez vous inscrire auprès de Red Hat, reportez-vous aux instructions dans Annexe C, *Création d'un compte Red Hat*.

```
[student@workstation ~]$ podman login registry.redhat.io
Username: your-username
Password: your-password
Login Succeeded!
```

- 2.2. Extrayez l'image de conteneur MySQL.

```
[student@workstation ~]$ podman pull registry.redhat.io/rhel8/mysql-80:1
Trying to pull registry.redhat.io/rhel8/mysql-80:1...
Getting image source signatures
Checking if image destination supports signatures
Copying blob 028bdc977650 done
...output omitted...
Writing manifest to image destination
Storing signatures
4ae3a3f4f409a8912cab9fbf71d3564d011ed2e68f926d50f88f2a3a72c809c5
```

- 2.3. Créez un conteneur en spécifiant le point de montage pour stocker les données de la base de données MySQL :

```
[student@workstation ~]$ podman run --name persist-db \
> -d -v /home/student/local/mysql:/var/lib/mysql/data \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
6e0ef134315b510042ca757faf869f2ba19df27790c601f95ec2fd9d3c44b95d
```

Cette commande monte le répertoire /home/student/local/mysql à partir de l'hôte dans le répertoire /var/lib/mysql/data du conteneur. Par défaut, la base

**chapitre 3 |** Gestion des conteneurs

de données MySQL stocke les données consignées dans le répertoire /var/lib/mysql/data.

2.4. Vérifiez que le conteneur a été correctement démarré.

```
[student@workstation ~]$ podman ps --format="{{.ID}} {{.Names}} {{.Status}}"
6e0ef134315b persist-db Up 3 minutes ago
```

► 3. Vérifiez que le répertoire /home/student/local/mysql/contient le répertoire items.

```
[student@workstation ~]$ ls -ld /home/student/local/mysql/items
drwxr-x---. 2 100026 100026 6 Apr 8 07:31 /home/student/local/mysql/items
```

Le répertoire items stocke les données liées à la base de données items créée par ce conteneur. Si le répertoire items n'est pas disponible, cela signifie que le point de montage n'a pas été correctement défini lors de la création du conteneur.

**Note**

Vous pouvez également exécuter la même commande avec podman unshare pour vérifier l'ID d'utilisateur (UID) numérique du conteneur.

```
[student@workstation ~]$ podman unshare ls -ld /home/student/local/mysql/items
drwxr-x---. 2 27 27 6 Apr 8 07:31 /home/student/local/mysql/items
```

## Fin

Sur workstation, exécutez le script lab manage-storage finish pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab manage-storage finish
```

Cela met fin à cet exercice.

# Accès aux conteneurs

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Décrire les notions de base de la mise en réseau avec des conteneurs.
- Se connecter à distance aux services à l'intérieur d'un conteneur.

## Mise en correspondance des ports réseau

L'accès à un conteneur sans racine à partir du réseau hôte peut être difficile, car aucune adresse IP n'est disponible pour un conteneur sans racine.

Pour résoudre ces problèmes, définissez des règles de transfert de port afin de permettre un accès externe à un service de conteneur. Utilisez l'option `-p [<IP address>:]<host port>:<container port>` avec la commande `podman run` pour créer un conteneur accessible depuis l'extérieur.

Prenons l'exemple suivant :

```
[user@host ~]$ podman run -d --name apache1 -p 8080:8080 \
> registry.redhat.io/rhel8/httpd-24
```

Cet exemple crée un conteneur accessible depuis l'extérieur. La valeur 8080 deux-points 8080 spécifie que toute demande au port 8080 sur l'hôte est transmise au port 8080 dans le conteneur.

Vous pouvez également utiliser l'option `-p` pour lier le port à l'adresse IP spécifiée.

```
[user@host ~]$ podman run -d --name apache2 \
> -p 127.0.0.1:8081:8080 registry.redhat.io/rhel8/httpd-24
```

Cet exemple limite l'accès externe au conteneur apache2 pour les demandes de `localhost` au port hôte 8081. Ces demandes sont transférées au port 8080 dans le conteneur apache2.

Si aucun port n'est spécifié pour le port hôte, Podman attribue un port hôte disponible aléatoire au conteneur :

```
[user@host ~]$ podman run -d --name apache3 -p 127.0.0.1::8080 \
> registry.redhat.io/rhel8/httpd-24
```

Pour voir le port attribué par Podman, utilisez la commande `podman port <container name>` :

```
[user@host ~]$ podman port apache3  
8080/tcp -> 127.0.0.1:35134  
  
[user@host ~]$ curl -s 127.0.0.1:35134 | egrep '</?title>'  
<title>Test Page for the HTTP Server on Red Hat Enterprise Linux</title>
```

Si seul un port de conteneur est spécifié avec l'option `-p`, alors un port d'hôte disponible aléatoire est attribué au conteneur. Les demandes adressées à ce port hôte attribué à partir de n'importe quelle adresse IP sont transférées au port de conteneur.

```
[user@host ~]$ podman run -d --name apache4 \  
> -p 8080 registry.redhat.io/rhel8/httpd-24  
  
[user@host ~]$ podman port apache4  
8080/tcp -> 0.0.0.0:37068
```

Dans cet exemple, toute demande routable vers le port hôte 37068 est transmise au port 8080 dans le conteneur.



## Références

**Container Network Interface - networking for Linux containers**

<https://github.com/containernetworking/cni>

**Cloud Native Computing Foundation**

<https://www.cncf.io/>

## ► Exercice guidé

# Chargement de la base de données

Dans cet exercice, vous allez créer un conteneur de base de données MySQL avec la fonction de transfert de port activée. Après avoir rempli une base de données avec un script SQL, vous vérifiez le contenu de la base de données à l'aide de trois méthodes différentes.

## Résultats

Vous devez pouvoir déployer un conteneur de base de données et charger un script SQL.

## Avant De Commencer

Ouvrez un terminal sur la machine `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab manage-networking start
```

Cela garantit que le répertoire `/home/student/local/mysql` existe et est configuré avec les autorisations appropriées pour activer le stockage persistant du conteneur MySQL.

## Instructions

- ▶ 1. Créez une instance de conteneur MySQL avec du stockage persistant et un transfert de port.
  - 1.1. Connectez-vous à Red Hat Container Catalog à l'aide de votre compte Red Hat. Si vous devez vous inscrire auprès de Red Hat, reportez-vous aux instructions dans Annexe C, *Création d'un compte Red Hat*.

```
[student@workstation ~]$ podman login registry.redhat.io
Username: your-username
Password: your-password
Login Succeeded!
```

- 1.2. Téléchargez l'image de conteneur de base de données MySQL, puis créez une instance de conteneur MySQL avec du stockage persistant et un transfert de port.

```
[student@workstation ~]$ podman run --name mysqladb-port \
> -d -v /home/student/local/mysql:/var/lib/mysql/data -p 13306:3306 \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
Trying to pull registry.redhat.io/rhel8/mysql-80:1...
Getting image source signatures
Checking if image destination supports signatures
Copying blob 0c673eb68f88 done
...output omitted...
```

```
Writing manifest to image destination
Storing signatures
066630d45cb902ab533d503c83b834aa6a9f9cf88755cb68eedb8a3e8edbc5aa
```

La dernière ligne de votre sortie et le temps nécessaire au téléchargement de chaque couche d'image seront différents. L'option `-p` configure le transfert de port de sorte que le port 13306 sur l'hôte local transfère au port de conteneur 3306.



### Note

Le script de démarrage crée le répertoire `/home/student/local/mysql` avec la propriété appropriée et le contexte SELinux requis par la base de données en conteneur.

- ▶ 2. Vérifiez que le conteneur `mysql ldb-port` a démarré avec succès et active le transfert de port.

```
[student@workstation ~]$ podman ps --format="{{.ID}} {{.Names}} {{.Ports}}"
9941da2936a5  mysql ldb-port  0.0.0.0:13306->3306/tcp
```

- ▶ 3. Renseignez la base de données à l'aide du fichier fourni. En l'absence d'erreur, la commande ci-dessus ne renvoie aucune sortie.

```
[student@workstation ~]$ mysql -uuser1 -h 127.0.0.1 -pmypassword -P13306 \
> items < /home/student/D0180/labs/manage-networking/db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
```

Il existe plusieurs façons de vérifier que la base de données a été correctement chargée. Les étapes suivantes illustrent trois méthodes différentes. Il vous suffit d'utiliser l'une des méthodes.

- ▶ 4. Vérifiez que la base de données a été correctement chargée en exécutant une commande non interactive à l'intérieur du conteneur.

```
[student@workstation ~]$ podman exec -it mysql ldb-port \
> mysql -uroot items -e "SELECT * FROM Item"
+-----+
| id | description      | done |
+-----+-----+-----+
| 1  | Pick up newspaper | 0   |
| 2  | Buy groceries     | 1   |
+-----+-----+-----+
```

- 5. Vérifiez que la base de données a été correctement chargée en utilisant le transfert de port à partir de l'hôte local. Cette autre méthode est facultative.

```
[student@workstation ~]$ mysql -uuser1 -h 127.0.0.1 -pmypa55 -P13306 \
> items -e "SELECT * FROM Item"
+---+-----+-----+
| id | description      | done |
+---+-----+-----+
| 1  | Pick up newspaper | 0   |
| 2  | Buy groceries     | 1   |
+---+-----+-----+
```

- 6. Vérifiez que la base de données a été correctement chargée en exécutant une session de terminal non interactive à l'intérieur du conteneur. Cette autre méthode est facultative.

- 6.1. Ouvrez un shell Bash à l'intérieur du conteneur.

```
[student@workstation ~]$ podman exec -it mysql-pod /bin/bash
bash-4.4$
```

- 6.2. Vérifiez que la base de données contient des données :

```
bash-4.4$ mysql -uroot items -e "SELECT * FROM Item"
+---+-----+-----+
| id | description      | done |
+---+-----+-----+
| 1  | Pick up newspaper | 0   |
| 2  | Buy groceries     | 1   |
+---+-----+-----+
```

- 6.3. Quittez le conteneur :

```
bash-4.4$ exit
exit
[student@workstation ~]$
```

## Fin

Sur `workstation`, exécutez le script `lab manage-networking finish` pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab manage-networking finish
```

Cela met fin à cet exercice.

## ► Open Lab

# Gestion des conteneurs

### Résultats

Vous devez pouvoir déployer et gérer une base de données persistante à l'aide d'un volume partagé. Vous devez également pouvoir démarrer une seconde base de données à l'aide d'un volume partagé et observer que les données sont cohérentes entre les deux conteneurs, car ces derniers utilisent le même répertoire sur l'hôte pour stocker les données MySQL.

### Avant De Commencer

Ouvrez un terminal sur `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab manage-review start
```

### Instructions

- Créez le répertoire `/home/student/local/mysql` avec les permissions et le contexte SELinux corrects.
- Déployez une instance de conteneur MySQL à l'aide de la spécification suivante :

Propriété	Description
Name	<code>mysql-1</code>
Run as daemon	<code>yes</code>
Volume	Lier le dossier hôte <code>/home/student/local/mysql</code> au dossier de conteneur <code>/var/lib/mysql/data</code>
Image de conteneur	<code>registry.redhat.io/rhel8/mysql-80:1</code>
Port forward	oui, lier le port hôte 13306 au port de conteneur 3306

Utilisez les variables d'environnement suivantes pour le conteneur :

Variable d'environnement	Valeur
<code>MYSQL_USER</code>	<code>user1</code>
<code>MYSQL_PASSWORD</code>	<code>mypa55</code>
<code>MYSQL_DATABASE</code>	<code>items</code>
<code>MYSQL_ROOT_PASSWORD</code>	<code>r00tpa55</code>

3. Chargez la base de données `items` à l'aide du script `/home/student/D0180/labs/manage-review/db.sql`.
4. Arrêtez correctement le conteneur.

**Important**

Cette étape est très importante, car un nouveau conteneur va être créé et partagera le même volume pour les données de la base de données. La base de données peut être endommagée lorsque deux conteneurs utilisent le même volume. Ne redémarrez pas le conteneur `mysql-1`.

5. Créez un conteneur avec la spécification suivante :

Propriété	Description
Name	<code>mysql-2</code>
Run as daemon	<code>yes</code>
Volume	Lier le dossier hôte <code>/home/student/local/mysql</code> au dossier de conteneur <code>/var/lib/mysql/data</code>
Image de conteneur	<code>registry.redhat.io/rhel8/mysql-80:1</code>
Port forward	oui, lier le port hôte 13306 au port de conteneur 3306

Utilisez les variables d'environnement suivantes pour le conteneur :

Variable d'environnement	Valeur
<code>MYSQL_USER</code>	<code>user1</code>
<code>MYSQL_PASSWORD</code>	<code>mypa55</code>
<code>MYSQL_DATABASE</code>	<code>items</code>
<code>MYSQL_ROOT_PASSWORD</code>	<code>r00tpa55</code>

6. Enregistrez la liste de tous les conteneurs (notamment les conteneurs arrêtés) dans le fichier `/tmp/my-containers`.
7. Accédez au shell Bash à l'intérieur du conteneur et vérifiez que la base de données `items` et la table `Item` sont toujours disponibles. Confirmez également que la table contient des données.
8. À l'aide du transfert de port, insérez une nouvelle ligne dans la table `Item`. La ligne doit comporter `description` pour la valeur `Finished lab` et `done` pour la valeur `1`.
9. Le premier conteneur n'étant plus requis, supprimez-le pour libérer les ressources.

## Évaluation

Notez votre travail en exécutant la commande `lab manage-review grade` à partir de votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab manage-review grade
```

## Fin

Sur workstation, exécutez la commande `lab manage-review finish` pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab manage-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Gestion des conteneurs

### Résultats

Vous devez pouvoir déployer et gérer une base de données persistante à l'aide d'un volume partagé. Vous devez également pouvoir démarrer une seconde base de données à l'aide d'un volume partagé et observer que les données sont cohérentes entre les deux conteneurs, car ces derniers utilisent le même répertoire sur l'hôte pour stocker les données MySQL.

### Avant De Commencer

Ouvrez un terminal sur **workstation** en tant qu'utilisateur **student** et exécutez la commande suivante :

```
[student@workstation ~]$ lab manage-review start
```

### Instructions

- Créez le répertoire `/home/student/local/mysql` avec les permissions et le contexte SELinux corrects.
  - Créez le répertoire `/home/student/local/mysql`.

```
[student@workstation ~]$ mkdir -vp /home/student/local/mysql
mkdir: created directory '/home/student/local/mysql'
```

- Ajoutez le contexte SELinux approprié pour le répertoire `/home/student/local/mysql` et son contenu. Avec le contexte correct, vous pouvez monter ce répertoire dans un conteneur en cours d'exécution.

```
[student@workstation ~]$ sudo semanage fcontext -a -t container_file_t \
> '/home/student/local/mysql(.*)?'
```

- Appliquez la politique SELinux au répertoire nouvellement créé.

```
[student@workstation ~]$ sudo restorecon -R /home/student/local/mysql
```

- Remplacez le propriétaire du répertoire `/home/student/local/mysql` pour qu'il corresponde à l'utilisateur `mysql` et au groupe `mysql` pour l'image de conteneur `registry.redhat.io/rhel8/mysql-80:1`:

```
[student@workstation ~]$ podman unshare chown -Rv 27:27 /home/student/local/mysql
changed ownership of '/home/student/local/mysql' from root:root to 27:27
```

- Déployez une instance de conteneur MySQL à l'aide de la spécification suivante :

Propriété	Description
Name	mysql-1
Run as daemon	yes
Volume	Lier le dossier hôte /home/student/local/mysql au dossier de conteneur /var/lib/mysql/data
Image de conteneur	registry.redhat.io/rhel8/mysql-80:1
Port forward	oui, lier le port hôte 13306 au port de conteneur 3306

Utilisez les variables d'environnement suivantes pour le conteneur :

Variable d'environnement	Valeur
MYSQL_USER	user1
MYSQL_PASSWORD	mypa55
MYSQL_DATABASE	items
MYSQL_ROOT_PASSWORD	r00tpa55

2.1. Connectez-vous à Red Hat Container Catalog à l'aide de votre compte Red Hat.

```
[student@workstation ~]$ podman login registry.redhat.io
Username: your-username
Password: your-password
Login Succeeded!
```

2.2. Créez et démarrez le conteneur.

```
[student@workstation ~]$ podman run --name mysql-1 -p 13306:3306 \
> -d -v /home/student/local/mysql:/var/lib/mysql/data \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
Trying to pull registry.redhat.io/rhel8/mysql-80:1...
Getting image source signatures
Checking if image destination supports signatures
Copying blob 0c673eb68f88 done
...output omitted...
Writing manifest to image destination
Storing signatures
6l6azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cdd
```

2.3. Vérifiez que le conteneur a été correctement démarré.

```
[student@workstation ~]$ podman ps --format="{{.ID}} {{.Names}}"
6l6azfaa55x8    mysql-1
```

**chapitre 3 |** Gestion des conteneurs

3. Chargez la base de données `items` à l'aide du script `/home/student/D0180/labs/manage-review/db.sql`.
  - 3.1. Chargez la base de données à l'aide des commandes SQL dans `/home/student/D0180/labs/manage-review/db.sql`.

```
[student@workstation ~]$ mysql -uuser1 -h 127.0.0.1 -pmypa55 -P13306 \
> items < /home/student/D0180/labs/manage-review/db.sql
mysql: [Warning] Using a password on the command-line interface can be insecure.
```

- 3.2. Utilisez une instruction SQL `SELECT` pour afficher toutes les lignes de la table `Item` afin de vérifier que la base de données `Items` est chargée.

**Note**

Vous pouvez ajouter le paramètre `-e SQL` à la commande `mysql` pour exécuter une instruction SQL.

```
[student@workstation ~]$ mysql -uuser1 -h 127.0.0.1 -pmypa55 -P13306 \
> items -e "SELECT * FROM Item"
+----+-----+----+
| id | description | done |
+----+-----+----+
| 1 | Pick up newspaper | 0 |
| 2 | Buy groceries | 1 |
+----+-----+----+
```

4. Arrêtez correctement le conteneur.

**Important**

Cette étape est très importante, car un nouveau conteneur va être créé et partagera le même volume pour les données de la base de données. La base de données peut être endommagée lorsque deux conteneurs utilisent le même volume. Ne redémarrez pas le conteneur `mysql-1`.

- 4.1. Utilisez la commande `podman` pour arrêter le conteneur.

```
[student@workstation ~]$ podman stop mysql-1
mysql-1
```

5. Créez un conteneur avec la spécification suivante :

Propriété	Description
Name	mysql-2
Run as daemon	yes
Volume	Lier le dossier hôte /home/student/local/mysql au dossier de conteneur /var/lib/mysql/data
Image de conteneur	registry.redhat.io/rhel8/mysql-80:1
Port forward	oui, lier le port hôte 13306 au port de conteneur 3306

Utilisez les variables d'environnement suivantes pour le conteneur :

Variable d'environnement	Valeur
MYSQL_USER	user1
MYSQL_PASSWORD	mypa55
MYSQL_DATABASE	items
MYSQL_ROOT_PASSWORD	r00tpa55

5.1. Créez et démarrez le conteneur.

```
[student@workstation ~]$ podman run --name mysql-2 -p 13306:3306 \
> -d -v /home/student/local/mysql:/var/lib/mysql/data \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
281c0e2790e54cd5a0b8e2a8cb6e3969981b85cde8ac611bf7ea98ff78bdffbb
```

5.2. Vérifiez que le conteneur a été correctement démarré.

```
[student@workstation ~]$ podman ps --format="{{.ID}} {{.Names}}"
281c0e2790e5    mysql-2
```

6. Enregistrez la liste de tous les conteneurs (notamment les conteneurs arrêtés) dans le fichier /tmp/my-containers.

6.1. Utilisez la commande podman pour enregistrer les informations dans /tmp/my-containers.

```
[student@workstation ~]$ podman ps -a > /tmp/my-containers
```

7. Accédez au shell Bash à l'intérieur du conteneur et vérifiez que la base de données items et la table Item sont toujours disponibles. Confirmez également que la table contient des données.

7.1. Accédez au shell Bash à l'intérieur du conteneur.

```
[student@workstation ~]$ podman exec -it mysql-2 /bin/bash
```

- 7.2. Connectez-vous au serveur MySQL.

```
bash-4.4$ mysql -uroot
Welcome to the MySQL monitor. Commands end with ; or \g.

...output omitted...

mysql>
```

- 7.3. Listez toutes les bases de données et confirmez que la base de données `items` est disponible.

```
mysql> SHOW DATABASES;
+-----+
| Database      |
+-----+
| information_schema |
| items          |
| mysql          |
| performance_schema |
| sys            |
+-----+
5 rows in set (0.03 sec)
```

- 7.4. Listez toutes les tables à partir de la base de données `items` et vérifiez que la table `Item` est disponible.

```
mysql> USE items;
Database changed

mysql> SHOW TABLES;
+-----+
| Tables_in_items |
+-----+
| Item           |
+-----+
1 row in set (0.01 sec)
```

- 7.5. Affichez les données à partir de la table.

```
mysql> SELECT * FROM Item;
+---+-----+---+
| id | description      | done |
+---+-----+---+
| 1  | Pick up newspaper |  0  |
| 2  | Buy groceries     |  1  |
+---+-----+---+
```

- 7.6. Quittez le client MySQL et le shell du conteneur.

**chapitre 3 |** Gestion des conteneurs

```
mysql> exit
Bye
bash-4.4$ exit
exit
[student@workstation ~]$
```

8. À l'aide du transfert de port, insérez une nouvelle ligne dans la table `Item`. La ligne doit comporter `description` pour la valeur `Finished lab` et `done` pour la valeur `1`.

8.1. Connectez-vous à la base de données MySQL.

```
[student@workstation ~]$ mysql -uuser1 -h workstation.lab.example.com \
> -pmypa55 -P13306 items
...output omitted...

Welcome to the MySQL monitor. Commands end with ; or \g.
...output omitted...

mysql>
```

8.2. Insérez la nouvelle ligne.

```
mysql> INSERT INTO Item (description, done) VALUES ('Finished lab', 1);
Query OK, 1 row affected (0.00 sec)
```

8.3. Quittez le client MySQL.

```
mysql> exit
Bye
[student@workstation ~]$
```

9. Le premier conteneur n'étant plus requis, supprimez-le pour libérer les ressources.

9.1. Utilisez la commande suivante pour supprimer le conteneur :

```
[student@workstation ~]$ podman rm mysql-1
6l6azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cdd
```

## Évaluation

Notez votre travail en exécutant la commande `lab manage-review grade` à partir de votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab manage-review grade
```

## Fin

Sur `workstation`, exécutez la commande `lab manage-review finish` pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab manage-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Podman a des sous-commandes pour : créer un nouveau conteneur (`run`), supprimer un conteneur (`rm`), lister des conteneurs (`ps`), arrêter un conteneur (`stop`) et démarrer un processus dans un conteneur (`exec`).
- Le stockage de conteneur par défaut est éphémère, ce qui signifie que son contenu n'est pas conservé après la suppression du conteneur.
- Les conteneurs peuvent utiliser un dossier du système de fichiers hôte pour utiliser des données persistantes.
- Podman monte des volumes dans un conteneur avec l'option `-v` dans la commande `podman run`.
- La commande `podman exec` démarre un processus supplémentaire dans un conteneur en cours d'exécution.
- Podman mappe les ports locaux sur les ports de conteneurs à l'aide de l'option `-p` dans la sous-commande `run`.

## chapitre 4

# Gestion des images de conteneur

### Objectif

Gérer le cycle de vie d'une image de conteneur, de la création à la suppression.

### Résultats

- Rechercher et extraire des images à partir de registres distants.
- Exporter, importer et gérer les images de conteneur localement et dans un registre.

### Sections

- Accès aux registres (et quiz)
- Manipulation d'images de conteneur (et exercice guidé)

### Atelier

- Gestion des images de conteneur

# Accès aux registres

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Rechercher et extraire des images de registres distants à l'aide de commandes Podman et de l'API REST du registre.
- Énumérer les avantages liés à l'utilisation d'un registre public certifié pour télécharger des images sécurisées.
- Personnaliser la configuration de Podman pour accéder à d'autres registres d'images de conteneur.
- Lister les images téléchargées à partir d'un registre vers le système de fichiers local.
- Gérer des balises pour extraire des images marquées.

## Registres publics

Les registres d'images sont des services proposant des images de conteneur à télécharger. Ils permettent aux créateurs et aux gestionnaires d'images de stocker et de distribuer des images de conteneur à un public ou privé.

Podman recherche et télécharge des images de conteneur à partir de registres publics et privés. Red Hat Container Catalog est le registre d'images public géré par Red Hat. Il héberge un grand nombre d'images de conteneur, y compris celles fournies par de grands projets Open Source, comme Apache, MySQL et Jenkins. Toutes les images de Container Catalog sont validées par l'équipe de sécurité interne de Red Hat et tous les composants ont été reconstruits par Red Hat pour éviter les vulnérabilités de sécurité connues.

Les images de conteneur Red Hat offrent les avantages suivants :

- *Source approuvée* : toutes les images de conteneur comprennent des sources connues et approuvées par Red Hat.
- *Dépendances originales* : aucun des paquetages de conteneurs n'a été falsifié, seules des bibliothèques connues sont incluses.
- *Sans vulnérabilité* : les images de conteneur sont exemptes des vulnérabilités connues dans les composants ou les couches de la plate-forme.
- *Protection d'exécution* : toutes les applications dans les images de conteneur s'exécutent en tant qu'utilisateurs non root, minimisant ainsi la surface d'exposition aux applications malveillantes ou défectueuses.
- *Compatibilité avec Red Hat Enterprise Linux (RHEL)* : les images de conteneur sont compatibles avec toutes les plateformes RHEL, des systèmes nus au cloud.
- *Services d'assistance Red Hat* : Red Hat prend commercialement en charge la pile complète.

Quay.io est un autre référentiel d'images public sponsorisé par Red Hat. Quay.io introduit plusieurs fonctionnalités intéressantes, telles que la génération d'images côté serveur, les contrôles d'accès précis et l'analyse automatique des images pour détecter les vulnérabilités connues.

Tandis que les images de Red Hat Container Catalog sont approuvées et vérifiées, Quay.io propose des images dynamiques régulièrement mises à jour par les créateurs. Les utilisateurs de Quay.io peuvent créer leurs espaces de noms, avec un contrôle d'accès précis, et publier les images qu'ils créent dans cet espace de noms. Les utilisateurs de Container Catalog envoient rarement (voire jamais) de nouvelles images, mais utilisent des images approuvées générées par l'équipe Red Hat.

## Registres privés

Les créateurs ou gestionnaires d'images souhaitent rendre leurs images publiques. Toutefois, d'autres créateurs d'images préfèrent toutefois garder leurs images privées pour les raisons suivantes :

- Politique de confidentialité de l'entreprise et protection des secrets.
- Restrictions légales et lois.
- Pour éviter de publier des images en cours de développement.

Dans certains cas, les images privées sont privilégiées. Les registres privés permettent aux créateurs d'images de contrôler leur placement, leur distribution et leur utilisation.

## Configuration des registres dans Podman



### Note

Le contenu suivant utilise le format de la version 1 pour la configuration du registre. Le format de la version 1 est obsolète dans les versions récentes de Podman et ne prend pas en charge l'utilisation de miroirs de registre, les correspondances du préfixe le plus long ni la réécriture d'emplacement.

Reportez-vous à la page de manuel `containers-registries.conf(5)` pour plus d'informations sur les fonctionnalités et le format de configuration de registre pris en charge.

Pour configurer des registres pour la commande `podman`, vous devez mettre à jour le fichier `/etc/containers/registries.conf`. Modifiez l'entrée `registries` dans la section `[registries.search]`, et ajoutez une entrée à la liste des valeurs :

```
[registries.search]
registries = ["registry.access.redhat.com", "quay.io"]
```



### Note

Utilisez un nom de domaine complet et un numéro de port pour identifier un registre. Un registre qui n'inclut pas de numéro de port a le numéro de port par défaut 5000. Si le registre utilise un port différent, vous devez le spécifier. Indiquez les numéros de port en ajoutant deux points (:) et le numéro de port après le nom de domaine complet.

Les connexions sécurisées à un registre nécessitent un certificat approuvé. Pour prendre en charge les connexions non sécurisées, ajoutez le nom du registre à l'entrée `registries` dans la section `[registries.insecure]` du fichier `/etc/containers/registries.conf`:

```
[registries.insecure]
registries = ['localhost:5000']
```

## Accès aux registres

Podman fournit des commandes qui vous permettent de rechercher des images. Les registres d'images sont également accessibles via une API. Les deux approches sont abordées ci-dessous.

### Recherche d'images dans les registres

La commande `podman search` recherche les images par nom d'image, nom d'utilisateur ou description dans tous les registres listés dans le fichier de configuration `/etc/containers/registries.conf`. La syntaxe de la commande `podman search` est affichée ci-dessous :

```
[user@host ~]$ podman search [OPTIONS] <term>
```

Le tableau suivant montre quelques-unes des options utiles disponibles pour la sous-commande `search` :

Option	Description
<code>--limit &lt;number&gt;</code>	Limite le nombre d'images listées par registre.
<code>--filter &lt;filter=value&gt;</code>	Filtrer la sortie en fonction des conditions fournies. Les filtres pris en charge sont les suivants : <code>stars=&lt;number&gt;</code> : afficher uniquement les images ayant au moins ce nombre d'étoiles. <code>is-automated=&lt;true false&gt;</code> : afficher uniquement les images générées automatiquement. <code>is-official=&lt;true false&gt;</code> : afficher uniquement les images marquées comme officielles.
<code>--tls-verify &lt;true false&gt;</code>	Active ou désactive la validation du certificat HTTPS pour tous les registres utilisés.
<code>--list-tags</code>	Liste les balises disponibles dans le référentiel pour l'image spécifiée.

### API HTTP du registre

Un registre distant expose des services Web fournissant une interface de programmation d'application (API) au registre. Podman utilise ces interfaces pour accéder aux référentiels distants et interagir avec eux. De nombreux registres sont conformes à la spécification Docker Registry HTTP API v2, qui expose une interface REST normalisée pour l'interaction avec le registre. Vous pouvez utiliser cette interface REST pour interagir directement avec un registre, au lieu d'utiliser Podman.

Des exemples utilisant cette API avec les commandes `curl` figurent ci-dessous :

Pour lister tous les référentiels disponibles dans un registre, utilisez le point d'accès `/v2/_catalog`. Le paramètre `n` est utilisé pour limiter le nombre de référentiels à renvoyer :

```
[user@host ~]$ curl -Ls https://myserver/v2/_catalog?n=3
>{"repositories":["centos/httpd","do180/custom-httpd","hello-openshift"]}
```



### Note

Si Python est disponible, utilisez-le pour formater la réponse JSON :

```
[user@host ~]$ curl -Ls https://myserver/v2/_catalog?n=3 \
> | python -m json.tool
{
  "repositories": [
    "centos/httpd",
    "do180/custom-httpd",
    "hello-openshift"
  ]
}
```

Le point d'accès /v2/<name>/tags/list fournit la liste des balises disponibles pour une seule image :

```
[user@host ~]$ curl -Ls \
> https://quay.io/v2/redhattraining/httpd-parent/tags/list \
> | python -m json.tool
{
  "name": "redhattraining/httpd-parent",
  "tags": [
    "latest",
    "2.4"
  ]
}
```



### Note

Quay.io propose une API dédiée pour interagir avec les référentiels, au-delà de ce qui est spécifié dans l'API Docker Repository. Consultez <https://docs.quay.io/api/> pour plus de détails.

## Authentification du registre

Certains registres d'images de conteneur nécessitent une autorisation d'accès. La commande `podman login` permet l'authentification du nom d'utilisateur et du mot de passe auprès d'un registre :

```
[user@host ~]$ podman login -u username \
> -p password registry.access.redhat.com
Login Succeeded!
```

L'API HTTP du registre nécessite des informations d'authentification. Tout d'abord, utilisez le service SSO (Single Sign On) de Red Hat pour obtenir un jeton d'accès :

```
[user@host ~]$ curl -u username:password -Ls \
> "https://sso.redhat.com/auth/realms/rhcc/protocol/redhat-docker-v2/auth?
service=docker-registry"
{"token":"eyJh...o5G8",
"access_token":"eyJh...mgL4",
"expires_in":...output omitted...}[user@host ~]$
```

Ensuite, incluez ce jeton dans un en-tête d'autorisation Bearer dans les demandes suivantes :

```
[user@host ~]$ curl -H "Authorization: Bearer eyJh...mgL4" \
> -Ls https://registry.redhat.io/v2/rhel8/mysql-80/tags/list \
> | python -mjson.tool
{
  "name": "rhel8/mysql-80",
  "tags": [
    "1.0",
    "1.2",
    ...output omitted...
```



### Note

D'autres registres peuvent nécessiter différentes étapes pour fournir les informations d'identification. Si un registre adhère à l'`Docker Registry HTTP v2 API`, l'authentification est conforme au schéma RFC7235.

## Extraction d'images

Pour extraire des images de conteneur d'un registre, utilisez la commande `podman pull`.

```
[user@host ~]$ podman pull [OPTIONS] [REGISTRY[:PORT]/]NAME[:TAG]
```

La commande `podman pull` utilise le nom d'image obtenu à partir de la sous-commande `search` pour extraire une image d'un registre. La sous-commande `pull` permet d'ajouter le nom du registre à l'image. Cette variante prend en charge la même image dans plusieurs registres.

Par exemple, pour extraire un conteneur NGINX du registre quay.io, utilisez la commande suivante :

```
[user@host ~]$ podman pull quay.io/bitnami/nginx
```

**Note**

Si le nom de l'image n'inclut pas de nom de registre, Podman recherche une image de conteneur correspondante à l'aide des registres listés dans le fichier de configuration `/etc/containers/registries.conf`. Podman recherche les images dans les registres dans le même ordre que celui où elles apparaissent dans le fichier de configuration.

Red Hat recommande de toujours inclure le nom du registre pour éviter l'utilisation accidentelle d'images inattendues.

Des utilisateurs malveillants peuvent télécharger des images dans des registres. Si ces registres apparaissent en premier dans votre configuration, vous utilisez l'image téléchargée par un acteur malveillant au lieu de celle que vous aviez l'intention d'utiliser.

## Lister les copies d'images locales

Toute image de conteneur téléchargée à partir d'un registre est stockée localement sur le même hôte sur lequel la commande `podman` est exécutée. Ce comportement évite de répéter les téléchargements d'images et réduit le temps de déploiement d'un conteneur. Podman stocke également toutes les images de conteneur personnalisées que vous construisez dans le même stockage local.

**Note**

Par défaut, Podman stocke les images de conteneur dans le répertoire `/var/lib/containers/storage/overlay-images`.

Podman fournit une sous-commande `images` pour lister toutes les images de conteneur stockées localement :

```
[user@host ~]$ podman images
REPOSITORY                                     TAG      IMAGE ID      CREATED       SIZE
registry.redhat.io/rhel8/mysql-80              latest   ad5a0e6d030f  3 weeks ago   588 MB
```

## Balises d'image

Une balise d'image est un mécanisme permettant de prendre en charge plusieurs versions de la même image. Cette fonction est utile lorsque plusieurs versions du même logiciel sont fournies, comme un conteneur prêt pour la production ou les dernières mises à jour du même logiciel développé pour l'évaluation de la communauté. Toute sous-commande Podman nécessitant un nom d'image de conteneur accepte un paramètre de balise pour différencier plusieurs balises. Si un nom d'image ne contient pas de balise, la valeur par défaut de la balise est `latest`. Par exemple, pour extraire une image avec la balise `1` à partir de `rhel8/mysql-80`, utilisez la commande suivante :

```
[user@host ~]$ podman pull registry.redhat.io/rhel8/mysql-80:1
```

Pour démarrer un nouveau conteneur basé sur l'image `rhel8/mysql-80:1`, utilisez la commande suivante :

```
[user@host ~]$ podman run registry.redhat.io/rhel8/mysql-80:1
```



## Références

### **Red Hat Container Catalog**

<https://registry.redhat.io>

### **Quay.io**

<https://quay.io>

### **Docker Registry HTTP API V2**

<https://github.com/docker/distribution/blob/master/docs/spec/api.md>

### **RFC7235 - HTTP/1.1: Authentication**

<https://tools.ietf.org/html/rfc7235>

## ► Quiz

# Utilisation des registres



### Note

Le questionnaire suivant utilise le format de la version 1 pour la configuration du registre. Le format de la version 1 est obsolète dans les versions récentes de Podman et ne prend pas en charge l'utilisation de miroirs de registre, les correspondances du préfixe le plus long ni la réécriture d'emplacement.

Reportez-vous à la page de manuel `containers-registries.conf(5)` pour plus d'informations sur les fonctionnalités et le format de configuration de registre pris en charge.

Répondez aux questions suivantes, en fonction des informations ci-après :

Podman est disponible sur un hôte RHEL avec l'entrée suivante dans le fichier `/etc/containers/registries.conf` :

```
[registries.search]
registries = ["registry.redhat.io", "quay.io"]
```

Les hôtes `registry.redhat.io` et `quay.io` ont un registre en cours d'exécution, les deux ont des certificats valides, et utilisent le registre de la version 1. Les images suivantes sont disponibles pour chaque hôte :

#### Noms d'image/balises par registre

Registre	Image
<code>registry.redhat.io</code>	<ul style="list-style-type: none"> <li>nginx:1.1.6</li> <li>mysql:8.0</li> <li>httpd:2.4</li> </ul>
<code>quay.io</code>	<ul style="list-style-type: none"> <li>mysql:5.7</li> <li>httpd:2.4</li> </ul>

Aucune image n'est disponible localement.

- 1. Quelles sont les deux commandes qui permettent d'afficher des images mysql disponibles au téléchargement à partir de `registry.redhat.io`? (Choisissez deux réponses.)
- `podman search registry.redhat.io/mysql`
  - `podman images`
  - `podman pull mysql`
  - `podman search mysql`

- 2. Quelle commande permet de lister toutes les balises d'image disponibles pour l'image de conteneur httpd ?
- a. podman search --list-tags httpd
  - b. podman images httpd
  - c. podman pull --all-tags=true httpd
  - d. Aucune commande podman n'est disponible pour rechercher des balises.
- 3. Quelles sont les deux commandes permettant d'extraire l'image httpd avec la balise 2.4 ? (Choisissez deux réponses.)
- a. podman pull httpd:2.4
  - b. podman pull httpd:latest
  - c. podman pull quay.io/httpd
  - d. podman pull registry.redhat.io/httpd:2.4
- 4. Lors de l'exécution des commandes suivantes, quelles images de conteneur seront téléchargées ?

```
[user@host ~]$ podman pull registry.redhat.io/httpd:2.4  
[user@host ~]$ podman pull quay.io/mysql:8.0
```

- a. quay.io/httpd:2.4, and `registry.redhat.io/mysql:8.0
- b. registry.redhat.io/httpd:2.4 et registry.redhat.io/mysql:8.0
- c. registry.redhat.io/httpd:2.4, aucune image ne sera téléchargée pour mysql.
- d. quay.io/httpd:2.4, aucune image ne sera téléchargée pour mysql.

## ► Solution

# Utilisation des registres



### Note

Le questionnaire suivant utilise le format de la version 1 pour la configuration du registre. Le format de la version 1 est obsolète dans les versions récentes de Podman et ne prend pas en charge l'utilisation de miroirs de registre, les correspondances du préfixe le plus long ni la réécriture d'emplacement.

Reportez-vous à la page de manuel `containers-registries.conf(5)` pour plus d'informations sur les fonctionnalités et le format de configuration de registre pris en charge.

Répondez aux questions suivantes, en fonction des informations ci-après :

Podman est disponible sur un hôte RHEL avec l'entrée suivante dans le fichier `/etc/containers/registries.conf` :

```
[registries.search]
registries = ["registry.redhat.io", "quay.io"]
```

Les hôtes `registry.redhat.io` et `quay.io` ont un registre en cours d'exécution, les deux ont des certificats valides, et utilisent le registre de la version 1. Les images suivantes sont disponibles pour chaque hôte :

#### Noms d'image/balises par registre

Registre	Image
<code>registry.redhat.io</code>	<ul style="list-style-type: none"> <li>• <code>nginx:1.1.6</code></li> <li>• <code>mysql:8.0</code></li> <li>• <code>httpd:2.4</code></li> </ul>
<code>quay.io</code>	<ul style="list-style-type: none"> <li>• <code>mysql:5.7</code></li> <li>• <code>httpd:2.4</code></li> </ul>

Aucune image n'est disponible localement.

- 1. Quelles sont les deux commandes qui permettent d'afficher des images `mysql` disponibles au téléchargement à partir de `registry.redhat.io`? (Choisissez deux réponses.)
- `podman search registry.redhat.io/mysql`
  - `podman images`
  - `podman pull mysql`
  - `podman search mysql`

- 2. Quelle commande permet de lister toutes les balises d'image disponibles pour l'image de conteneur httpd ?
- a. podman search --list-tags httpd
  - b. podman images httpd
  - c. podman pull --all-tags=true httpd
  - d. Aucune commande podman n'est disponible pour rechercher des balises.
- 3. Quelles sont les deux commandes permettant d'extraire l'image httpd avec la balise 2.4 ? (Choisissez deux réponses.)
- a. podman pull httpd:2.4
  - b. podman pull httpd:latest
  - c. podman pull quay.io/httpd
  - d. podman pull registry.redhat.io/httpd:2.4
- 4. Lors de l'exécution des commandes suivantes, quelles images de conteneur seront téléchargées ?

```
[user@host ~]$ podman pull registry.redhat.io/httpd:2.4  
[user@host ~]$ podman pull quay.io/mysql:8.0
```

- a. quay.io/httpd:2.4, and `registry.redhat.io/mysql:8.0
- b. registry.redhat.io/httpd:2.4 et registry.redhat.io/mysql:8.0
- c. registry.redhat.io/httpd:2.4, aucune image ne sera téléchargée pour mysql.
- d. quay.io/httpd:2.4, aucune image ne sera téléchargée pour mysql.

# Manipulation d'images de conteneur

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Enregistrer et charger les images de conteneur dans des fichiers locaux.
- Supprimer des images du stockage local.
- Créer de nouvelles images de conteneur à partir de conteneurs et mettre à jour les métadonnées de l'image.
- Gérer les balises d'image à des fins de distribution.

## Introduction

Il existe différentes manières de gérer les conteneurs d'images tout en respectant les principes DevOps. Par exemple, un développeur finit de tester un conteneur personnalisé sur une machine et il doit transférer cette image de conteneur vers un autre hôte pour un autre développeur, ou vers un serveur de production. Il y a deux manières de procéder :

1. Enregistrer l'image de conteneur dans un fichier .tar.
2. Publish (*push*) the container image to an image registry.



### Note

L'une des façons dont un développeur pourrait avoir créé ce conteneur personnalisé est décrite ultérieurement dans ce chapitre (`podman commit`). Cependant, dans les chapitres suivants, nous discutons de la manière recommandée de le faire en utilisant `Containerfiles`.

## Enregistrement et chargement d'images

Les images existantes du stockage local Podman peuvent être enregistrées dans un fichier .tar à l'aide de la commande `podman save`. Le fichier généré n'est pas une archive TAR ordinaire : il contient des métadonnées d'image et conserve les couches d'image d'origine. À l'aide de ce fichier, Podman peut recréer l'image d'origine.

La syntaxe générale de la sous-commande `save` est la suivante :

```
[user@host ~]$ podman save [-o FILE_NAME] IMAGE_NAME[:TAG]
```

Podman envoie l'image générée à la sortie standard sous forme de données binaires. Pour éviter cela, utilisez l'option `-o`.

L'exemple suivant enregistre l'image de conteneur MySQL téléchargée précédemment à partir de Red Hat Container Catalog dans le fichier `mysql.tar` :

```
[user@host ~]$ podman save \  
> -o mysql.tar registry.redhat.io/rhel8/mysql-80
```

Notez l'utilisation de l'option `-o` dans cet exemple.

Utilisez les fichiers `.tar` générés par la sous-commande `save` à des fins de sauvegarde. Pour restaurer l'image de conteneur, utilisez la commande `podman load`. La syntaxe générale de la commande est la suivante :

```
[user@host ~]$ podman load [-i FILE_NAME]
```

La commande `load` est le contraire de la commande `save`.

Par exemple, cette commande chargerait une image enregistrée dans un fichier nommé `mysql.tar` :

```
[user@host ~]$ podman load -i mysql.tar
```

Si le fichier `.tar` donné en argument n'est pas une image de conteneur avec des métadonnées, la commande `podman load` échoue.



#### Note

Pour économiser de l'espace disque, compressez le fichier généré par la sous-commande `save` avec Gzip en utilisant le paramètre `--compress`. La sous-commande `load` utilise la commande `gunzip` avant d'importer le fichier dans le stockage local.

## Suppression d'images

Podman conserve toutes les images téléchargées dans son stockage local, même celles qui ne sont actuellement pas utilisées par un conteneur. Cependant, les images peuvent devenir obsolètes et doivent être remplacées par la suite.



#### Note

Les mises à jour d'images dans un registre ne sont pas automatiquement mises à jour. L'image doit être supprimée, puis extraite à nouveau, pour garantir que le stockage local dispose de la dernière version de celle-ci.

Pour supprimer une image du cache local, exécutez la commande `podman rmi`.

```
[user@host ~]$ podman rmi [OPTIONS] IMAGE [IMAGE...]
```

Une image peut être référencée en utilisant le nom ou l'identifiant à des fins de suppression. Podman ne peut pas supprimer des images lorsque les conteneurs les utilisent. Vous devez arrêter et supprimer tous les conteneurs utilisant cette image avant de la supprimer.

Pour éviter cela, la sous-commande `rmi` offre l'option `--force`. Cette option force la suppression d'une image même si celle-ci est utilisée par plusieurs conteneurs ou si ces conteneurs sont en

cours d'exécution. Podman arrête et supprime tous les conteneurs à l'aide de l'image supprimée de force avant de la supprimer.

## Suppression de toutes les images

Pour supprimer toutes les images qui ne sont utilisées par aucun conteneur, utilisez la commande suivante :

```
[user@host ~]$ podman rmi -a
```

La commande renvoie tous les ID d'image disponibles dans le stockage local et les transmet en tant que paramètres à la commande `podman rmi` pour suppression. Les images en cours d'utilisation ne sont pas supprimées. Toutefois, cela n'empêche pas l'élimination des images inutilisées.

## Modification d'images

Idéalement, toutes les images de conteneur doivent être générées en utilisant un `Containerfile` pour créer un ensemble propre et léger de couches d'image, sans fichiers journaux, fichiers temporaires ou autres artefacts créés par la personnalisation du conteneur. Cependant, certains utilisateurs peuvent fournir des images de conteneur telles quelles, sans aucun `Containerfile`. Au lieu de créer des images, modifiez un conteneur en cours d'exécution sur place et enregistrez ses couches pour créer une image de conteneur. La commande `podman commit` fournit cette fonction.



### Mise en garde

Bien que la commande `podman commit` représente la manière la plus directe de créer des images, elle n'est pas recommandée en raison de la taille des images (`commit` conserve les journaux et les fichiers d'ID de processus dans les couches capturées) et du manque de traçabilité des modifications. `Containerfile` fournit un mécanisme robuste pour personnaliser et implémenter les modifications apportées à un conteneur à l'aide d'un ensemble de commandes lisibles par un humain, sans l'ensemble de fichiers générés par le système d'exploitation.

La syntaxe de la commande `podman commit` est la suivante :

```
[user@host ~]$ podman commit [OPTIONS] CONTAINER \
> [REPOSITORY[:PORT]/]IMAGE_NAME[:TAG]
```

Le tableau suivant montre les options les plus importantes disponibles pour la commande `podman commit` :

Option	Description
<code>--author ""</code>	Identifie celui qui a créé l'image de conteneur.
<code>--message ""</code>	Inclut un message de validation dans le registre.
<code>--format</code>	Sélectionne le format de l'image. Les options valides sont oci et docker.

**Note**

L'option `--message` n'est pas disponible dans le format de conteneur OCI par défaut.

Pour trouver l'ID d'un conteneur en cours d'exécution dans Podman, exécutez la commande `podman ps` :

```
[user@host ~]$ podman ps
CONTAINER ID IMAGE ...
87bdfcc7c656 mysql ...output omitted... mysql-basic
```

À terme, les administrateurs peuvent personnaliser l'image et définir le conteneur à l'état souhaité. Pour identifier les fichiers qui ont été modifiés, créés ou supprimés depuis le démarrage du conteneur, utilisez la sous-commande `diff`. Cette sous-commande requiert uniquement le nom du conteneur ou l'ID du conteneur :

```
[user@host ~]$ podman diff mysql-basic
C /run
C /run/mysqld
A /run/mysqld/mysqld.pid
A /run/mysqld/mysqld.sock
A /run/mysqld/mysqld.sock.lock
A /run/secrets
```

La sous-commande `diff` marque tous les fichiers ajoutés avec un A, tous les fichiers modifiés avec un C et tous les fichiers supprimés avec un D.

**Note**

La commande `diff` ne signale au système de fichiers du conteneur que les fichiers qui ont été ajoutés, modifiés ou supprimés. Les fichiers qui sont montés sur un conteneur en cours d'exécution ne sont pas considérés comme faisant partie du système de fichiers du conteneur. Pour récupérer la liste des fichiers et répertoires montés pour un conteneur en cours d'exécution, utilisez la commande `podman inspect` :

```
[user@host ~]$ podman inspect \
> -f "{{range .Mounts}}{{println .Destination}}{{end}}" CONTAINER_NAME/ID
```

Les fichiers figurant dans cette liste ou sous un répertoire de cette liste, ne s'affichent pas dans la sortie de la commande `podman diff`.

Pour valider les modifications sur une autre image, exécutez la commande suivante :

```
[user@host ~]$ podman commit mysql-basic mysql-custom
```

L'exemple précédent crée une image nommée `mysql-custom`.

## Balisage d'images

Un projet avec plusieurs images basées sur le même logiciel peut être distribué, créant ainsi des projets individuels pour chaque image, mais, cette approche nécessite plus de maintenance pour gérer et déployer les images aux emplacements corrects.

Les registres d'images de conteneur prennent en charge les balises pour pouvoir distinguer plusieurs versions du même projet. Par exemple, un client peut utiliser une image de conteneur à exécuter avec une base de données MySQL ou PostgreSQL, en utilisant une balise pour différencier la base de données qui doit être utilisée par une image de conteneur.



### Note

Habituellement, les balises sont utilisées par les développeurs de conteneurs pour distinguer plusieurs versions du même logiciel. Plusieurs balises sont fournies pour identifier une version facilement. Le site Web officiel de l'image de conteneur MySQL utilise la version comme nom de la balise (8.0). De plus, la même image peut posséder une deuxième balise avec la version inférieure pour réduire la nécessité d'obtenir la dernière édition d'une version spécifique.

Pour baliser une image, utilisez la commande `podman tag` :

```
[user@host ~]$ podman tag [OPTIONS] IMAGE[:TAG] \
> [REGISTRYHOST/] [USERNAME/]NAME[:TAG]
```

L'argument `IMAGE` correspond au nom d'image avec une balise facultative gérée par Podman. L'argument suivant fait référence au nouveau nom alternatif pour l'image. Podman suppose qu'il s'agit de la dernière version, comme indiqué par la balise `latest`, si la valeur de balise est absente. Par exemple, pour marquer une image, utilisez la commande suivante :

```
[user@host ~]$ podman tag mysql-custom devops/mysql
```

L'option `mysql-custom` correspond au nom d'image dans le registre de conteneur.

Pour utiliser un nom de balise différent, utilisez plutôt la commande suivante :

```
[user@host ~]$ podman tag mysql-custom devops/mysql:snapshot
```

## Suppression des balises dans des images

Une seule image peut avoir plusieurs balises attribuées à l'aide de la commande `podman tag`. Pour les supprimer, utilisez la commande `podman rmi`, comme mentionné précédemment :

```
[user@host ~]$ podman rmi devops/mysql:snapshot
```



### Note

Étant donné que plusieurs balises peuvent pointer vers la même image, pour supprimer une image référencée par plusieurs balises, supprimez d'abord chacune d'entre elles.

## Meilleures pratiques pour le balisage d'images

Podman ajoute automatiquement la balise `latest` si vous ne spécifiez aucune balise, parce que Podman considère que l'image correspond à la dernière compilation. Cependant, cela peut varier selon la façon dont chaque projet utilise les balises. Par exemple, la plupart des projets Open Source considèrent que la balise `latest` correspond à la mouture la plus récente, pas à la dernière build.

De plus, plusieurs balises sont fournies pour réduire le besoin de se rappeler de la dernière édition d'une version particulière d'un projet. Ainsi, s'il existe une version de projet, par exemple `2.1.10`, une autre balise nommée `2.1` peut être créée pour pointer vers la même image à partir de la version `2.1.10`. Cela simplifie l'extraction des images du registre.

## Publication d'images dans un registre

Pour publier une image dans un registre, elle doit se trouver dans le stockage local de Podman et être balisée à des fins d'identification. Pour pousser l'image vers le registre, utilisez la sous-commande `push` :

```
[user@host ~]$ podman push [OPTIONS] IMAGE [DESTINATION]
```

Par exemple, pour envoyer l'image `bitnami/nginx` (par `push`) vers son référentiel, utilisez la commande suivante :

```
[user@host ~]$ podman push quay.io/bitnami/nginx
```

L'exemple précédent pousse l'image vers Quay.io.



### Références

#### Site Podman

<https://podman.io/>

## ► Exercice guidé

# Création d'une image de conteneur Apache personnalisée

Au cours de cet exercice guidé, vous allez créer une image de conteneur Apache personnalisée à l'aide de la commande `podman commit`.

## Résultats

Vous devez pouvoir créer une image de conteneur personnalisée.

## Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Ouvrez un terminal sur la machine `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab image-operations start
```

## Instructions

- ▶ 1. Connectez-vous à votre compte Quay.io et démarrez un conteneur à l'aide de l'image disponible à l'adresse `quay.io/redhattraining/httpd-parent`. L'option `-p` vous permet de spécifier un port de redirection. Dans ce cas, Podman achemine les demandes entrantes sur le port TCP 8180 de l'hôte vers le port TCP 80 du conteneur.

```
[student@workstation ~]$ podman login quay.io
Username: your_quay_username
Password: your_quay_password
Login Succeeded!

[student@workstation ~]$ podman run -d --name official-httpd \
> -p 8180:80 quay.io/redhattraining/httpd-parent
Trying to pull quay.io/redhattraining/httpd-parent:latest...
Getting image source signatures
Copying blob a3ed95caeb02 done
...output omitted...
Writing manifest to image destination
Storing signatures
3a6baecaff2b4e8c53b026e04847dda5976b773ade1a3a712b1431d60ac5915d
```

Votre dernière ligne de sortie est différente de la dernière ligne affichée ci-dessus. Notez les douze premiers caractères.

- ▶ 2. Créez une page HTML dans le conteneur `official-httpd`.

**chapitre 4 |** Gestion des images de conteneur

- 2.1. Accédez au shell du conteneur en utilisant la commande `podman exec`, puis créez une page HTML.

```
[student@workstation ~]$ podman exec -it official-httdp /bin/bash  
bash-4.4# echo "D0180 Page" > /var/www/html/do180.html
```

- 2.2. Quittez le conteneur.

```
bash-4.4# exit  
exit
```

- 2.3. Vérifiez que le fichier HTML est accessible à partir de la machine `workstation` en utilisant la commande `curl`.

```
[student@workstation ~]$ curl 127.0.0.1:8180/do180.html
```

La sortie attendue est similaire à la suivante :

```
D0180 Page
```

- 3. Utilisez la commande `podman diff` pour examiner les différences dans le conteneur entre l'image et la nouvelle couche créée par ce dernier.

```
[student@workstation ~]$ podman diff official-httdp  
C /etc  
C /root  
A /root/.bash_history  
...output omitted...  
C /tmp  
C /var  
C /var/log  
C /var/log/httdp  
A /var/log/httdp/access_log  
A /var/log/httdp/error_log  
C /var/www  
C /var/www/html  
A /var/www/html/do180.html
```

**Note**

Souvent, les images du conteneur de serveur Web identifient le répertoire `/var/www/html` en tant que volume. Dans ces cas, tous les fichiers ajoutés à ce répertoire ne sont pas considérés comme faisant partie du système de fichiers du conteneur, et ne s'afficheraient pas dans la sortie de la commande `podman diff`.

L'image de conteneur `quay.io/redhat/training/httdp-parent` n'identifie pas le répertoire `/var/www/html` en tant que volume. À ce titre, la modification apportée au fichier `/var/www/html/do180.html` est considérée comme une modification apportée au système de fichiers du conteneur sous-jacent.

**chapitre 4 |** Gestion des images de conteneur

- 4. Créez une nouvelle image avec les modifications créées par le conteneur en cours d'exécution.

- 4.1. Arrêtez le conteneur **official-httdp**.

```
[student@workstation ~]$ podman stop official-httdp  
official-httdp
```

- 4.2. Validez les modifications apportées à une nouvelle image de conteneur avec un nouveau nom. Utilisez votre nom comme auteur des modifications.

```
[student@workstation ~]$ podman commit -a 'Your Name' \  
> official-httdp do180-custom-httdp  
Getting image source signatures  
...output omitted...  
Copying blob 7f9108fde4a1 skipped: already exists  
Copying blob 4c0f28c7efd0 done  
Copying config 5c2bb39b76 done  
Writing manifest to image destination  
Storing signatures  
31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb399befef2a23
```

- 4.3. Listez les images de conteneur disponibles.

```
[student@workstation ~]$ podman images
```

La sortie attendue est similaire à la suivante :

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/do180-custom-httdp	latest	<b>31c3ac78e9d4</b>	...	...
quay.io/redhattraining/httpd-parent	latest	2cc07fbb5000	...	...

L'ID de l'image correspond au 12 premiers caractères du hachage. Les images les plus récentes figurent en haut de la liste.

- 5. Publiez l'image de conteneur enregistrée dans le registre de conteneur.

- 5.1. Chargez la configuration de votre environnement de formation.

Exécutez la commande suivante pour charger les variables d'environnement créées lors du premier exercice guidé :

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 5.2. Pour marquer l'image avec la balise et le nom d'hôte du registre, exécutez la commande suivante.

```
[student@workstation ~]$ podman tag do180-custom-httdp \  
> quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
```

- 5.3. Exécutez la commande **podman images** pour vous assurer que le nouveau nom a été ajouté au cache.

```
[student@workstation ~]$ podman images
REPOSITORY                                     TAG      IMAGE ID      ...
localhost/do180-custom-httdp                 latest   31c3ac78e9d4  ...
quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp v1.0    31c3ac78e9d4  ...
quay.io/redhattraining/httpd-parent          latest   2cc07fbb5000  ...
```

5.4. Publiez l'image dans votre registre Quay.io.

```
[student@workstation ~]$ podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
Getting image source signatures
...output omitted...
Copying blob 24d85c895b6b skipped: already exists
Copying config 5c2bb39b76 done
Writing manifest to image destination
Storing signatures
```



### Note

L'envoi de l'image `do180-custom-httdp` par push crée un référentiel privé dans votre compte Quay.io. Actuellement, les référentiels privés ne sont pas autorisés par les plans gratuits Quay.io. Vous pouvez soit créer le référentiel public avant d'envoyer l'image par push, soit définir le référentiel sur « public » par la suite.

5.5. Vérifiez que l'image est disponible à partir de Quay.io. La commande `podman search` nécessite que l'image soit indexée par Quay.io. Cela peut prendre quelques heures. Dès lors, utilisez la commande `podman pull` pour récupérer l'image. Cela prouve que l'image est disponible.

```
[student@workstation ~]$ podman pull \
> -q quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeab3
```

► 6. Créez un conteneur à partir de l'image récemment publiée.

Utilisez la commande `podman run` pour démarrer un nouveau conteneur. Utilisez `your_quay_username/do180-custom-httdp:v1.0` comme image de base.

```
[student@workstation ~]$ podman run -d --name test-httdp -p 8280:80 \
> ${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
c0f04e906bb12bd0e514cbd0e581d2746e04e44a468dfbc85bc29ffcc5acd16c
```

► 7. Utilisez la commande `curl` pour accéder à la page HTML. Veillez à utiliser le port 8280.

Cela devrait afficher la page HTML créée à l'étape précédente.

```
[student@workstation ~]$ curl http://localhost:8280/do180.html
DO180 Page
```

## Fin

Sur `workstation`, exéutez le script `lab image-operations finish` pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab image-operations finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Gestion des images

### Résultats

Vous devez pouvoir créer une image de conteneur personnalisée et gérer les images de conteneur.

### Avant De Commencer

Ouvrez un terminal sur `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab image-review start
```

### Instructions

1. Utilisez la commande `podman pull` pour télécharger l'image de conteneur `quay.io/redhattraining/nginx:1.17`. Cette image est une copie de l'image de conteneur officielle disponible sur `docker.io/library/nginx:1.17`. Assurez-vous que vous avez correctement téléchargé l'image.
2. Démarrez un nouveau conteneur au moyen de l'image Nginx, conformément aux spécifications figurant dans la liste suivante.

Propriété	Description
Name	<code>official-nginx</code>
Run as daemon	<code>yes</code>
Image de conteneur	<code>quay.io/redhattraining/nginx:1.17</code>
Port forward	oui, du port hôte 8080 au port de conteneur 80.

3. Connectez-vous au conteneur à l'aide de la commande `podman exec`. Remplacez le contenu du fichier `index.html` par `D0180`. Le répertoire du serveur Web se trouve sur `/usr/share/nginx/html`. Une fois que le fichier a été mis à jour, quittez le conteneur et utilisez la commande `curl` pour accéder à la page Web.
4. Arrêtez le conteneur en cours d'exécution et validez vos modifications pour créer une nouvelle image de conteneur. Donnez à la nouvelle image le nom `do180/mynginx` et la balise `v1.0-SNAPSHOT`. Utilisez les spécifications suivantes :

Propriété	Description
Nom de l'image	do180/mynginx
Image tag	v1.0-SNAPSHOT
Author name	<i>otre nom</i>

5. Démarrez un nouveau conteneur au moyen de l'image Nginx mise à jour, conformément aux spécifications figurant dans la liste suivante.

Propriété	Description
Name	official-nginx-dev
Run as daemon	yes
Image de conteneur	do180/mynginx:v1.0-SNAPSHOT
Port forward	oui, du port hôte 8080 au port de conteneur 80.

6. Connectez-vous au conteneur à l'aide de la commande `podman exec`, puis apportez une dernière modification. Remplacez le contenu du fichier `/usr/share/nginx/html/index.html` par D0180 Page.
- Une fois que le fichier a été mis à jour, quittez le conteneur et utilisez la commande `curl` pour vérifier les modifications.
7. Arrêtez le conteneur en cours d'exécution et validez vos modifications pour créer l'image de conteneur finale. Donnez à la nouvelle image le nom `do180/mynginx` et la balise `v1.0`. Utilisez les spécifications suivantes :

Propriété	Description
Nom de l'image	do180/mynginx
Image tag	v1.0
Author name	<i>otre nom</i>

8. Supprimez l'image de développement `do180/mynginx:v1.0-SNAPSHOT` à partir du stockage d'images local.

9. Utilisez l'image marquée `do180/mynginx:v1.0` pour créer un nouveau conteneur avec les spécifications suivantes :

Propriété	Description
Name	<code>my-nginx</code>
Run as daemon	<code>yes</code>
Image de conteneur	<code>do180/mynginx:v1.0</code>
Port forward	oui, du port hôte 8280 au port de conteneur 80.

Sur `workstation`, utilisez la commande `curl` pour accéder au serveur Web, accessible à partir du port 8280.

## Évaluation

Notez votre travail en exécutant la commande `lab image-review grade` sur votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab image-review grade
```

## Fin

Sur `workstation`, exécutez la commande `lab image-review finish` pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab image-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Gestion des images

### Résultats

Vous devez pouvoir créer une image de conteneur personnalisée et gérer les images de conteneur.

### Avant De Commencer

Ouvrez un terminal sur `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab image-review start
```

### Instructions

- Utilisez la commande `podman pull` pour télécharger l'image de conteneur `quay.io/redhattraining/nginx:1.17`. Cette image est une copie de l'image de conteneur officielle disponible sur `docker.io/library/nginx:1.17`.

Assurez-vous que vous avez correctement téléchargé l'image.

- Utilisez la commande `podman pull` pour extraire l'image de conteneur Nginx.

```
[student@workstation ~]$ podman pull quay.io/redhattraining/nginx:1.17
Trying to pull quay.io/redhattraining/nginx:1.17...
...output omitted...
Storing signatures
9beeba249f3ee158d3e495a6ac25c5667ae2de8a43ac2a8bfd2bf687a58c06c9
```

- Vérifiez que l'image de conteneur existe en local en exécutant la commande `podman images`.

```
[student@workstation ~]$ podman images
```

Cette commande génère un résultat similaire à celui présenté ci-dessous :

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
quay.io/redhattraining/nginx	1.17	9beeba249f3e	6 months ago	428MB

- Démarrez un nouveau conteneur au moyen de l'image Nginx, conformément aux spécifications figurant dans la liste suivante.

Propriété	Description
Name	official-nginx
Run as daemon	yes
Image de conteneur	quay.io/redhat/training/nginx:1.17
Port forward	oui, du port hôte 8080 au port de conteneur 80.

- 2.1. Sur **workstation**, utilisez la commande `podman run` pour créer un conteneur nommé **official-nginx**.

```
[student@workstation ~]$ podman run --name official-nginx \
> -d -p 8080:80 quay.io/redhat/training/nginx:1.17
b9d5739af239914b371025c38340352ac1657358561e7ebbd5472dfd5ff97788
```

3. Connectez-vous au conteneur à l'aide de la commande `podman exec`. Remplacez le contenu du fichier `index.html` par D0180. Le répertoire du serveur Web se trouve sur `/usr/share/nginx/html`.

Une fois que le fichier a été mis à jour, quittez le conteneur et utilisez la commande `curl` pour accéder à la page Web.

- 3.1. Connectez-vous au conteneur à l'aide de la commande `podman exec`.

```
[student@workstation ~]$ podman exec -it official-nginx /bin/bash
root@b9d5739af239:/#
```

- 3.2. Mettez à jour le fichier `index.html` situé à l'adresse `/usr/share/nginx/html`. Le fichier doit indiquer D0180.

```
root@b9d5739af239:/# echo 'D0180' > /usr/share/nginx/html/index.html
```

- 3.3. Quittez le conteneur.

```
root@b9d5739af239:/# exit
exit
```

- 3.4. Utilisez la commande `curl` pour vous assurer que le fichier `index.html` est mis à jour.

```
[student@workstation ~]$ curl 127.0.0.1:8080
D0180
```

4. Arrêtez le conteneur en cours d'exécution et validez vos modifications pour créer une nouvelle image de conteneur. Donnez à la nouvelle image le nom `do180/mynginx` et la balise `v1.0-SNAPSHOT`. Utilisez les spécifications suivantes :

**chapitre 4 |** Gestion des images de conteneur

Propriété	Description
Nom de l'image	do180/mynginx
Image tag	v1.0-SNAPSHOT
Author name	votre nom

- 4.1. Utilisez la commande `podman stop` pour arrêter le conteneur `official-nginx`.

```
[student@workstation ~]$ podman stop official-nginx
official-nginx
```

- 4.2. Validez vos modifications apportées à une nouvelle image de conteneur. Utilisez votre nom comme auteur des modifications.

```
[student@workstation ~]$ podman commit -a 'Your Name' \
> official-nginx do180/mynginx:v1.0-SNAPSHOT
Getting image source signatures
...output omitted...
Copying blob 1e69eb596a89 done
Copying config e8c32f0b53 done
Writing manifest to image destination
Storing signatures
d6d10f52e258e4e88c181a56c51637789424e9261b208338404e82a26c960751
```

- 4.3. Listez les images de conteneur disponibles pour localiser l'image que vous avez créée récemment.

```
[student@workstation ~]$ podman images
REPOSITORY          TAG      IMAGE ID      CREATED       ...
localhost/do180/mynginx    v1.0-SNAPSHOT  d6d10f52e258  30 seconds ago ...
quay.io/redhattraining/nginx  1.17      9beeba249f3e  6 months ago ...
```

5. Démarrez un nouveau conteneur au moyen de l'image Nginx mise à jour, conformément aux spécifications figurant dans la liste suivante.

Propriété	Description
Name	official-nginx-dev
Run as daemon	yes
Image de conteneur	do180/mynginx:v1.0-SNAPSHOT
Port forward	oui, du port hôte 8080 au port de conteneur 80.

- 5.1. Sur `workstation`, utilisez la commande `podman run` pour créer un conteneur nommé `official-nginx-dev`.

**chapitre 4** | Gestion des images de conteneur

```
[student@workstation ~]$ podman run --name official-nginx-dev \
> -d -p 8080:80 do180/mynginx:v1.0-SNAPSHOT
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a28866156671f0bbbb
```

6. Connectez-vous au conteneur à l'aide de la commande `podman exec`, puis apportez une dernière modification. Remplacez le contenu du fichier `/usr/share/nginx/html/index.html` par D0180 Page.

Une fois que le fichier a été mis à jour, quittez le conteneur et utilisez la commande `curl` pour vérifier les modifications.

- 6.1. Connectez-vous au conteneur à l'aide de la commande `podman exec`.

```
[student@workstation ~]$ podman exec -it official-nginx-dev /bin/bash
root@cfa21f02a77d:/#
```

- 6.2. Mettez à jour le fichier `index.html` situé à l'adresse `/usr/share/nginx/html`. Le fichier doit indiquer D0180 Page.

```
root@cfa21f02a77d:/# echo 'D0180 Page' > /usr/share/nginx/html/index.html
```

- 6.3. Quittez le conteneur.

```
root@cfa21f02a77d:/# exit
exit
```

- 6.4. Utilisez la commande `curl` pour vous assurer que le fichier `index.html` est mis à jour.

```
[student@workstation ~]$ curl 127.0.0.1:8080
D0180 Page
```

7. Arrêtez le conteneur en cours d'exécution et validez vos modifications pour créer l'image de conteneur finale. Donnez à la nouvelle image le nom `do180/mynginx` et la balise `v1.0`. Utilisez les spécifications suivantes :

Propriété	Description
Nom de l'image	<code>do180/mynginx</code>
Image tag	<code>v1.0</code>
Author name	<i>votre nom</i>

- 7.1. Utilisez la commande `podman stop` pour arrêter le conteneur `official-nginx-dev`.

```
[student@workstation ~]$ podman stop official-nginx-dev
official-nginx-dev
```

- 7.2. Validez vos modifications apportées à une nouvelle image de conteneur. Utilisez votre nom comme auteur des modifications.

**chapitre 4 |** Gestion des images de conteneur

```
[student@workstation ~]$ podman commit -a 'Your Name' \
> official-nginx-dev do180/mynginx:v1.0
Getting image source signatures
...output omitted...
Copying blob 26d691d5f9e8 done
Copying config 60332a1074 done
Writing manifest to image destination
Storing signatures
90915976c33de534e06778a74d2a8969c25ef5f8f58c0c1ab7aeaac19abd18af
```

- 7.3. Listez les images de conteneur disponibles pour localiser votre image récemment créée.

```
[student@workstation ~]$ podman images
REPOSITORY           TAG      IMAGE ID      CREATED     ...
localhost/do180/mynginx    v1.0      90915976c33d   6 seconds ago ...
localhost/do180/mynginx    v1.0-SNAPSHOT  d6d10f52e258   8 minutes ago ...
quay.io/redhattraining/nginx  1.17      9beeba249f3e   6 months ago ...
```

8. Supprimez l'image de développement do180/mynginx:v1.0-SNAPSHOT à partir du stockage d'images local.

- 8.1. Malgré qu'il a été arrêté, `official-nginx-dev` est toujours présent. Affichez le conteneur avec la commande `podman ps` à l'aide de l'indicateur `-a`.

```
[student@workstation ~]$ podman ps -a --format="{{.ID}} {{.Names}} {{.Status}}"
cfa21f02a77d  official-nginx-dev  Exited (0) 9 minutes ago
b9d5739af239  official-nginx      Exited (0) 12 minutes ago
```

- 8.2. Supprimez le conteneur avec la commande `podman rm`.

```
[student@workstation ~]$ podman rm official-nginx-dev
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a288661566671f0bbbb
```

- 8.3. Vérifiez que le conteneur est supprimé en soumettant à nouveau la même commande `podman ps`.

```
[student@workstation ~]$ podman ps -a --format="{{.ID}} {{.Names}} {{.Status}}"
b9d5739af239  official-nginx      Exited (0) 12 minutes ago
```

- 8.4. Utilisez la commande `podman rmi` pour supprimer l'image `do180/mynginx:v1.0-SNAPSHOT`.

```
[student@workstation ~]$ podman rmi do180/mynginx:v1.0-SNAPSHOT
Untagged: localhost/do180/mynginx:v1.0-SNAPSHOT
```

- 8.5. Vérifiez que l'image n'est plus présente en répertoriant toutes les images à l'aide de la commande `podman images`.

```
[student@workstation ~]$ podman images
REPOSITORY           TAG      IMAGE ID      CREATED       SIZE
localhost/do180/mynginx   v1.0    90915976c33d  5 minutes ago  131MB
quay.io/redhattraining/nginx  1.17    9beeba249f3e  6 months ago  131MB
```

9. Utilisez l'image marquée `do180/mynginx:v1.0` pour créer un nouveau conteneur avec les spécifications suivantes :

Propriété	Description
Name	<code>my-nginx</code>
Run as daemon	<code>yes</code>
Image de conteneur	<code>do180/mynginx:v1.0</code>
Port forward	oui, du port hôte 8280 au port de conteneur 80.

Sur `workstation`, utilisez la commande `curl` pour accéder au serveur Web, accessible à partir du port 8280.

- 9.1. Utilisez la commande `podman run` pour créer le conteneur `my-nginx`, conformément aux spécifications.

```
[student@workstation ~]$ podman run -d --name my-nginx \
> -p 8280:80 do180/mynginx:v1.0
51958c8ec8d2613bd26f85194c66ca96c95d23b82c43b23b0f0fb9fded74da20
```

- 9.2. Utilisez la commande `curl` pour vérifier que la page `index.html` est disponible et renvoie le contenu personnalisé.

```
[student@workstation ~]$ curl 127.0.0.1:8280
DO180 Page
```

## Évaluation

Notez votre travail en exécutant la commande `lab image-review grade` sur votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab image-review grade
```

## Fin

Sur `workstation`, exécutez la commande `lab image-review finish` pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab image-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Red Hat Container Catalog fournit des images testées et certifiées sur [registry.redhat.io](https://registry.redhat.io).
- Podman peut interagir avec des registres de conteneurs distants pour rechercher, extraire et pousser des images de conteneur.
- Les balises d'image sont un mécanisme permettant de prendre en charge plusieurs versions d'une image de conteneur.
- Podman fournit des commandes permettant de gérer les images de conteneur à la fois dans un stockage local et sous forme de fichiers compressés.
- Utilisez la commande `podman commit` pour créer une image à partir d'un conteneur.



## chapitre 5

# Création d'images de conteneur personnalisées

### Objectif

Concevoir et coder un Containerfile pour créer une image de conteneur personnalisée.

### Résultats

- Décrire les approches de création des images de conteneur personnalisées.
- Créer une image de conteneur à l'aide des commandes Containerfile courantes.

### Sections

- Conception d'images de conteneur personnalisées (avec quiz)
- Création d'images de conteneur personnalisées avec des Containerfiles (avec exercice guidé)

### Atelier

- Création d'images de conteneur personnalisées

# Conception d'images de conteneur personnalisées

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Décrire les approches de création des images de conteneur personnalisées.
- Identifier les Containerfile existants à utiliser comme point de départ pour créer une image de conteneur personnalisée.
- Définir le rôle joué par Red Hat Software Collections Library (RHSCl) dans la conception des images de conteneur à partir du registre Red Hat.
- Décrire l'autre méthode, à savoir S2I (Source-to-Image), par rapport aux Containerfiles.

## Réutilisation des Containerfiles existants

Une méthode de création d'images de conteneur décrite précédemment nécessite de créer un conteneur, de le modifier pour répondre aux exigences de l'application à exécuter dans celui-ci, puis de valider les modifications apportées à une image. Cette option, bien que simple, convient uniquement à l'utilisation ou au test de modifications très spécifiques. Elle ne respecte pas les meilleures pratiques logicielles, telles que la capacité de maintenance, l'automatisation de la compilation et la répétabilité.

Les Containerfiles constituent une autre option permettant de créer des images de conteneur répondant à ces limitations. Les Containerfiles sont faciles à partager, à réutiliser et à étendre. Il est également facilement d'en contrôler les versions.

Les Containerfiles facilitent également l'extension d'une image, appelée une image enfant, à partir d'une autre image, appelée une image parente. Une image enfant intègre tout ce qui se trouve dans l'image parente, ainsi que tous les changements et ajouts apportés pour la créer.



### Note

Pour ce reste de ce cours, un fichier appelé Containerfile peut être un fichier nommé *Containerfile* ou *Dockerfile*. Les deux partagent la même syntaxe. *Containerfile* est le nom par défaut utilisé par les outils compatibles OCI.

Pour partager et réutiliser des images, de nombreuses applications, langages et structures populaires sont déjà disponibles dans des registres d'images publics, tels que Quay.io [<https://quay.io>]. La personnalisation de la configuration d'une application pour se conformer aux pratiques recommandées relatives aux conteneurs n'est pas une tâche facile. C'est pourquoi en partant d'une image parente éprouvée, vous déployez généralement nettement moins d'efforts.

L'utilisation d'une image parente de grande qualité améliore les possibilités de maintenance, en particulier si l'image parente est constamment mise à jour par son auteur, afin d'appliquer les correctifs de bogues et de sécurité.

La création d'un Containerfile pour construire une image enfant à partir d'une image de conteneur existante est souvent utilisée dans l'un des scénarios typiques suivants :

- l'ajout de nouvelles bibliothèques d'exécution, telles que les connecteurs de base de données ;
- l'inclusion de personnalisations à l'échelle de l'organisation, telles que les certificats SSL et les fournisseurs d'authentification ;
- l'ajout de bibliothèques internes partageables par plusieurs images de conteneur en tant que couche d'image unique pour différentes applications ;

la modification d'un Containerfile existant pour créer une image peut également être une approche raisonnable dans d'autres scénarios. Par exemple :

- Réduisez l'image de conteneur en supprimant les éléments inutilisés (tels que les pages de manuel ou la documentation figurant dans /usr/share/doc).
- Verrouillez l'image parent ou un paquetage logiciel inclus sur une version spécifique afin de réduire les risques liés aux futures mises à jour logicielles.

Docker Hub et Red Hat Software Collections Library (RHSC) sont deux sources d'images de conteneur à utiliser, en tant qu'images parentes ou pour modifier les Containerfiles.

## Utilisation de Red Hat Software Collections Library

Red Hat Software Collections Library (RHSC), ou plus simplement Software Collections, est la solution Red Hat pour les développeurs ayant besoin des outils de développement les plus récents qui sont généralement publiés en dehors du calendrier standard de publication de RHEL.

Red Hat Enterprise Linux (RHEL) offre un environnement stable pour les applications d'entreprise. Pour cela, RHEL doit conserver les versions importantes des paquetages en amont au même niveau afin d'empêcher les modifications de format de fichier de configuration et d'API. Les correctifs de sécurité et de performance sont rétroportés des dernières versions en amont, mais les nouvelles fonctionnalités qui ne respecteraient pas la compatibilité ascendante ne le sont pas.

RHSC permet aux développeurs de logiciels d'utiliser la version la plus récente sans incidence sur RHEL, car les paquetages RHSC ne remplacent ni n'entrent en conflit avec les paquetages RHEL par défaut. Les paquetages RHEL par défaut et les paquetages RHSC sont installés côte à côte.



### Note

Tous les abonnés RHEL ont accès à RHSC. To enable a particular software collection for a specific user or application environment (for example, MySQL 5.7, which is named rh-mysql57), enable the RHSC software Yum repositories and follow a few simple steps.

## Identification des Containerfiles dans Red Hat Software Collections Library

RHSC est la source de la plupart des images de conteneur fournies par le registre d'images Red Hat et utilisables par les clients RHEL Atomic Host et OpenShift Container Platform.

Red Hat fournit les Containerfiles RHSC et autres sources connexes dans le paquetage `rhscl-dockerfiles` disponible dans le référentiel RHSC. Les utilisateurs de la communauté peuvent obtenir les Containerfiles des images de conteneur comparables à celles de CentOS à partir du référentiel GitHub sur <https://github.com/sclorg?q=-container>.

**Note**

De nombreuses images de conteneur RHSCl incluent la prise en charge de Source-to-Image (S2I), mieux connue en tant que fonction OpenShift Container Platform. La prise en charge de S2I n'a aucune incidence sur l'utilisation de ces images de conteneur avec Docker.

## Images de conteneur dans Red Hat Container Catalog (RHCC)

Les applications critiques requièrent des conteneurs approuvés. Red Hat Container Catalog est un référentiel de la collection fiable, testée, certifiée et supervisée d'images de conteneur, compilé à partir de versions de Red Hat Enterprise Linux (RHEL) et de systèmes connexes. Les images de conteneur disponibles via RHCC ont fait l'objet d'un processus d'assurance qualité. Tous les composants ont été recompilés par Red Hat pour éviter les vulnérabilités de sécurité connues. Ils sont mis à niveau régulièrement de façon à contenir la version requise du logiciel, même lorsqu'une nouvelle image n'est pas encore disponible. À l'aide de RHCC, vous pouvez parcourir et rechercher des images, et vous pouvez accéder aux informations relatives à chaque image, telles que sa version, son contenu et son utilisation.

## Recherche d'images à l'aide de Quay.io

Quay.io est un référentiel de conteneurs avancé de CoreOS optimisé pour la collaboration en équipe. Vous pouvez rechercher les images de conteneur à l'aide de <https://quay.io/search>.

En cliquant sur le nom d'une image, vous accédez à la page d'informations sur l'image, y compris à toutes les balises existantes pour l'image, ainsi qu'à la commande permettant d'extraire l'image.

## Identification des Containerfiles sur Docker Hub

Faites attention aux images de Docker Hub. Toute personne peut créer un compte Docker Hub et publier des images de conteneur sur le site. Il n'existe aucune garantie de qualité ni de sécurité ; les images publiées sur Docker Hub peuvent avoir été créées par des professionnels ou être le résultat d'expérimentations ponctuelles. Vous devez évaluer chaque image de manière individuelle.

La recherche d'une image aboutit à la page de documentation susceptible de fournir le lien vers le Containerfile correspondant. Par exemple, le premier résultat de la recherche de mysql est la page de documentation de l'image MySQL official dans [https://hub.docker.com/\\_/mysql](https://hub.docker.com/_/mysql).

Sur cette page, sous la section `Supported tags and respective Dockerfile links`, chacune des balises pointe vers le projet GitHub `docker-library`, qui héberge les Containerfiles des images créées par le système de build automatique de la communauté Docker.

L'URL directe de l'arborescence des Containerfiles de Docker Hub MySQL 8.0 est <https://github.com/docker-library/mysql/blob/master/8.0>.

## Description de l'utilisation de l'outil Source-to-Image d'OpenShift

Source-to-Image (S2I) permet de créer des images de conteneur indépendamment des Containerfiles. Il peut être utilisé comme une fonctionnalité OpenShift ou en tant qu'utilitaire s2i autonome. S2I permet aux développeurs de travailler avec leurs propres outils. Ils n'ont pas besoin

## chapitre 5 | Création d'images de conteneur personnalisées

de connaître la syntaxe Containerfile et utilisent des commandes de système d'exploitation telles que yum. Généralement, l'utilitaire S2I crée des images de taille inférieure, avec moins de couches.

S2I utilise le processus suivant pour créer une image de conteneur personnalisée pour une application :

1. Commencez à créer un conteneur à partir d'une image de conteneur de base désignée comme image d'outil de création. Cette image comprend un programme d'exécution de langage de programmation et des outils essentiels de développement, tels que des compilateurs et des gestionnaires de paquetages.
2. Récupérez le code source de l'application, généralement à partir du serveur Git, puis envoyez-le au conteneur.
3. Générez les fichiers binaires d'application à l'intérieur du conteneur.
4. Après avoir procédé au nettoyage, enregistrez le conteneur en tant que nouvelle image de conteneur comprenant le programme d'exécution du langage de programmation et des fichiers binaires de l'application.

L'image d'outil de création est une image de conteneur normale respectant la structure standard d'un répertoire et fournissant des scripts qui sont appelés au cours du processus S2I. Vous pouvez utiliser la plupart de ces images d'outil de création en tant qu'images de base pour les Containerfiles, en dehors du processus S2I.

La commande s2i sert à exécuter le processus S2I en dehors d'OpenShift, dans un environnement Docker unique. Vous pouvez l'installer sur un système RHEL à partir du paquetage RPM source-to-image et sur d'autres plateformes, notamment Windows et Mac OS, à partir des programmes d'installation disponibles dans le projet S2I sur GitHub.



### Références

#### **Red Hat Software Collections Library (RHSCl)**

<https://access.redhat.com/documentation/en/red-hat-software-collections/>

#### **Red Hat Container Catalog (RHCC)**

<https://access.redhat.com/containers/>

#### **Dockerfiles RHSCl sur GitHub**

<https://github.com/sclorg?q=-container>

#### **Utilisation d'images de conteneur Red Hat Software Collections**

<https://access.redhat.com/articles/1752723>

#### **Quay.io**

<https://quay.io/search>

#### **Docker Hub**

<https://hub.docker.com/>

#### **Projet GitHub de la bibliothèque Docker**

<https://github.com/docker-library>

#### **Projet GitHub S2I**

[https://github.com/openshift/source-to-image/](https://github.com/openshift/source-to-image)

## ► Quiz

# Méthodes relatives à la conception d'images de conteneur

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

► 1. **Quelle est la méthode de création d'images de conteneur recommandée par la communauté de conteneurs ?**

- a. Exécutez les commandes à l'intérieur d'un conteneur de système d'exploitation de base, validez le conteneur, puis enregistrez-le ou exportez-le en tant que nouvelle image de conteneur.
- b. Exécutez les commandes à partir d'un Containerfile et envoyez l'image de conteneur générée dans un registre d'images.
- c. Créez manuellement les couches d'image de conteneur à partir de fichiers tar.
- d. Exécutez la commande `podman build` pour traiter la description de l'image de conteneur au format YAML.

► 2. **Parmi les options suivantes, laquelle est un avantage de l'utilisation du processus S2I autonome par rapport aux Containerfiles ?**

- a. N'exige aucun outil supplémentaire hormis la configuration Podman de base.
- b. Crée des images de conteneur de plus petite taille avec moins de couches.
- c. Réutilise des images d'outil de création de grande qualité.
- d. Met automatiquement à jour l'image enfant via des modifications de l'image parente (par exemple, avec des correctifs de sécurité).
- e. Crée des images compatibles avec OpenShift, contrairement aux images de conteneur créées à partir des outils Docker.

► 3. **Quels sont les deux scénarios classiques permettant de créer un Containerfile afin de générer une image enfant à partir d'une image existante ? (Choisissez deux réponses.)**

- a. L'ajout de nouvelles bibliothèques d'exécution.
- b. Définition de contraintes pour l'accès sur un conteneur au processeur de la machine hôte.
- c. Ajout de bibliothèques internes pouvant être partagées par plusieurs images de conteneur en tant qu'une couche d'image unique pour différentes applications.

## ► Solution

# Méthodes relatives à la conception d'images de conteneur

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- 1. **Quelle est la méthode de création d'images de conteneur recommandée par la communauté de conteneurs ?**
- a. Exécutez les commandes à l'intérieur d'un conteneur de système d'exploitation de base, validez le conteneur, puis enregistrez-le ou exportez-le en tant que nouvelle image de conteneur.
  - b. Exécutez les commandes à partir d'un Containerfile et envoyez l'image de conteneur générée dans un registre d'images.
  - c. Créez manuellement les couches d'image de conteneur à partir de fichiers tar.
  - d. Exécutez la commande podman build pour traiter la description de l'image de conteneur au format YAML.
- 2. **Parmi les options suivantes, laquelle est un avantage de l'utilisation du processus S2I autonome par rapport aux Containerfiles ?**
- a. N'exige aucun outil supplémentaire hormis la configuration Podman de base.
  - b. Crée des images de conteneur de plus petite taille avec moins de couches.
  - c. Réutilise des images d'outil de création de grande qualité.
  - d. Met automatiquement à jour l'image enfant via des modifications de l'image parente (par exemple, avec des correctifs de sécurité).
  - e. Crée des images compatibles avec OpenShift, contrairement aux images de conteneur créées à partir des outils Docker.
- 3. **Quels sont les deux scénarios classiques permettant de créer un Containerfile afin de générer une image enfant à partir d'une image existante ? (Choisissez deux réponses.)**
- a. L'ajout de nouvelles bibliothèques d'exécution.
  - b. Définition de contraintes pour l'accès sur un conteneur au processeur de la machine hôte.
  - c. Ajout de bibliothèques internes pouvant être partagées par plusieurs images de conteneur en tant qu'une couche d'image unique pour différentes applications.

# Compilation d'images de conteneur personnalisées avec des Containerfiles

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure de créer une image de conteneur à l'aide des commandes Containerfile courantes.

## Compilation de conteneurs de base

Un Containerfile est un mécanisme permettant d'automatiser la création d'images de conteneur. La création d'une image à partir d'un Containerfile est un processus en trois étapes :

1. Créer un répertoire de travail
2. Écrire le Containerfile
3. Compiler l'image avec Podman

## Créer un répertoire de travail

Le répertoire de travail est le répertoire contenant tous les fichiers nécessaires à la génération de l'image. La création d'un répertoire de travail vide est une bonne pratique pour éviter d'incorporer des fichiers inutiles dans l'image. Pour des raisons de sécurité, n'utilisez jamais le répertoire root, /, en tant que répertoire de travail pour la création d'images.

## Écrire la spécification Containerfile

Un Containerfile est un fichier texte appelé *Containerfile* ou *Dockerfile*, qui comprend les instructions nécessaire pour compiler l'image. La syntaxe de base d'un Containerfile est la suivante :

```
# Comment
INSTRUCTION arguments
```

Les lignes commençant par le symbole dièse (#) sont des commentaires. *INSTRUCTION* fait référence à tout mot-clé d'instruction Containerfile. Les instructions ne sont pas sensibles à la casse mais, par convention, les instructions sont mises en majuscules pour améliorer leur visibilité.

La première instruction (autre que des commentaires) doit être **FROM** pour indiquer l'image de base. Les instructions Containerfile sont exécutées dans un nouveau conteneur à l'aide de cette image, puis validées dans une nouvelle image. La prochaine instruction (le cas échéant) s'exécute dans cette nouvelle image. L'ordre d'exécution des instructions est identique à l'ordre dans lequel elles apparaissent dans le Containerfile.



### Note

L'instruction ARG peut apparaître devant l'instruction FROM, mais les instructions ARG sont en dehors des objectifs de cette section.

Chaque instruction Containerfile s'exécute dans un conteneur indépendant à l'aide d'une image intermédiaire créée à partir de chaque commande précédente. Cela signifie que chaque instruction est indépendante des autres instructions du Containerfile. Voici un exemple de Containerfile permettant de créer un conteneur de serveur Web Apache simple :

```
# This is a comment line ①
FROM      ubi8/ubi:8.5 ②
LABEL     description="This is a custom httpd container image" ③
MAINTAINER John Doe <jdoe@xyz.com> ④
RUN       yum install -y httpd ⑤
EXPOSE   80 ⑥
ENV       LogLevel "info" ⑦
ADD       http://someserver.com/filename.pdf /var/www/html ⑧
COPY      ./src/  /var/www/html/ ⑨
USER     apache ⑩
ENTRYPOINT ["/usr/sbin/httpd"] ⑪
CMD      ["-D", "FOREGROUND"] ⑫
```

- ① Les lignes qui commencent par le signe « dièse » (#) sont des commentaires.
- ② L'instruction FROM déclare que la nouvelle image de conteneur étend l'image de base du conteneur `ubi8/ubi:8.5`. Les Containerfiles peuvent utiliser n'importe quelle autre image de conteneur en tant qu'image de base, et pas seulement les images provenant des distributions du système d'exploitation. Red Hat fournit un ensemble d'images de conteneur qui sont certifiées et testées, et recommande vivement d'utiliser ces images de conteneur comme base.
- ③ LABEL est responsable de l'ajout de métadonnées génériques à une image. Un LABEL est une paire clé-valeur simple.
- ④ MAINTAINER indique le champ Author des métadonnées de l'image de conteneur générée. Vous pouvez utiliser la commande `podman inspect` pour afficher les métadonnées de l'image.
- ⑤ RUN exécute des commandes dans une nouvelle couche au-dessus de l'image en cours. Le shell utilisé pour exécuter les commandes est `/bin/sh`.
- ⑥ EXPOSE indique que le conteneur écoute sur le port réseau spécifié au moment de l'exécution. L'instruction EXPOSE définit uniquement des métadonnées. Elle ne rend pas les ports accessibles depuis l'hôte. L'option `-p` dans la commande `podman run` expose les ports de conteneur de l'hôte.
- ⑦ ENV est responsable de la définition des variables d'environnement qui sont disponibles dans le conteneur. Vous pouvez déclarer plusieurs instructions ENV dans le Containerfile. Vous pouvez utiliser la commande `env` dans le conteneur pour afficher chacune des variables d'environnement.
- ⑧ L'instruction ADD copie les fichiers ou les dossiers à partir de la source locale ou distante et les ajoute au système de fichiers du conteneur. Si elle est utilisée pour copier des fichiers locaux, ceux-ci doivent se trouver dans le répertoire de travail. L'instruction ADD décomprime les fichiers `.tar` locaux dans le répertoire des images de destination.
- ⑨ COPY copie les fichiers à partir du répertoire de travail et les ajoute au système de fichiers du conteneur. Il n'est pas possible de copier un fichier distant en utilisant son URL avec cette instruction Containerfile.

- ⑩ USER spécifie le nom d'utilisateur ou l'UID à utiliser lors de l'exécution de l'image de conteneur pour les instructions RUN, CMD et ENTRYPPOINT. Il est recommandé de définir un autre utilisateur que root pour des raisons de sécurité.
- ⑪ ENTRYPPOINT spécifie la commande par défaut à exécuter lorsque l'image s'exécute dans un conteneur. Si la valeur est omise, l'ENTRYPPOINT par défaut est /bin/sh -c.
- ⑫ CMD fournit les arguments par défaut pour l'instruction ENTRYPPOINT. Si l'ENTRYPPOINT par défaut s'applique (/bin/sh -c), alors CMD forme une commande exécutable et des paramètres qui s'exécutent au démarrage du conteneur.

## CMD et ENTRYPPOINT

Les instructions ENTRYPPOINT et CMD présentent deux formats :

### Forme exécutable

Cette forme utilise un tableau JSON :

```
ENTRYPPOINT ["command", "param1", "param2"]
CMD           ["param1", "param2"]
```

### Forme shell

```
ENTRYPPOINT command param1 param2
CMD           param1  param2
```

La forme exécutable est la forme préférée. La forme shell enveloppe les commandes dans un shell /bin/sh -c, créant un processus shell parfois inutile. En outre, certaines combinaisons ne sont pas autorisées ou risquent de ne pas fonctionner comme prévu. Par exemple, si ENTRYPPOINT correspond à ["ping"] (forme exécutable) et CMD correspond à localhost (forme shell), alors la commande exécutée attendue est ping localhost, mais le conteneur essaie ping /bin/sh -c localhost, qui est une commande malformée.

Le Containerfile doit contenir au moins un ENTRYPPOINT et une instruction CMD. Si plusieurs instructions redondantes sont présentes, seule la dernière instruction prend effet. CMD peut être présent sans spécifier d'ENTRYPPOINT. Dans ce cas, l'ENTRYPPOINT de l'image de base s'applique, ou l'ENTRYPPOINT par défaut si aucun n'est défini.

Podman peut ignorer l'instruction CMD lors du démarrage d'un conteneur. Le cas échéant, tous les paramètres de la commande podman run après le nom de l'image forment l'instruction CMD. Par exemple, l'instruction suivante entraîne l'affichage de l'heure actuelle par le conteneur en cours d'exécution.

```
ENTRYPPOINT ["/bin/date", "+%H:%M"]
```

L'ENTRYPPOINT définit à la fois la commande à exécuter et les paramètres. L'instruction CMD ne peut dès lors pas être utilisée. L'exemple suivant fournit la même fonctionnalité que la précédente avec, en outre, l'avantage que l'instruction CMD est remplacable lorsqu'un conteneur est démarré.

```
ENTRYPPOINT ["/bin/date"]
CMD           ["+"%H:%M"]
```

Dans les deux cas, lorsqu'un conteneur démarre sans fournir de paramètre, l'heure actuelle est affichée :

```
[student@workstation ~]$ sudo podman run -it do180/rhel  
11:41
```

Dans le deuxième cas, si un paramètre apparaît après le nom de l'image dans la commande `podman run`, il remplace l'instruction CMD. La commande suivante affiche le jour actuel de la semaine au lieu de l'heure :

```
[student@workstation demo-basic]$ sudo podman run -it do180/rhel +%A  
Tuesday
```

Une autre approche utilise l'ENTRYPOINT par défaut et l'instruction CMD pour définir la commande initiale. L'instruction suivante affiche l'heure actuelle, avec l'avantage supplémentaire de pouvoir être remplacée au moment de l'exécution.

```
CMD ["date", "+%H:%M"]
```

## ADD et COPY

Les instructions ADD et COPY présentent deux formes :

### Forme shell

```
ADD source ... destination  
COPY source ... destination
```

### Forme exécutable :

```
ADD ["source", ... "destination"]  
COPY ["source", ... "destination"]
```

Si la source est un chemin de système de fichiers, il doit se trouver dans le répertoire de travail.

L'instruction ADD vous permet également de spécifier une ressource en utilisant une URL.

```
ADD http://some server.com/filename.pdf /var/www/html
```

Si la source est une archive tar locale dans un format de compression reconnu (identity, gzip, bzip2 ou xz), l'instruction ADD décomprime l'archive en tant que répertoire dans le dossier de *destination*. L'instruction COPY n'a pas cette fonctionnalité.

**Important**

Les instructions ADD et COPY copient les fichiers normaux, avec `root` comme propriétaire. L'instruction ADD récupère le contenu de l'URL et le fichier est copié avec `root` comme propriétaire. L'instruction ADD extrait l'archive tar et conserve le propriétaire ainsi que les autorisations.

Vous pouvez utiliser l'option `--chown=<user>:<group>` avec la commande COPY ou ADD pour définir le propriétaire souhaité ou utiliser une instruction RUN après la copie afin de définir le propriétaire et les autorisations.

## Superposition d'images

Chaque instruction dans un Containerfile crée une nouvelle couche d'image. Un trop grand nombre d'instructions dans un Containerfile crée trop de couches, et donne lieu à de grandes images. Prenons par exemple les instructions RUN suivantes dans un Containerfile :

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms"
RUN yum update -y
RUN yum install -y httpd
```

Cet exemple montre la création d'une image de conteneur comportant trois couches (une pour chaque instruction RUN). Red Hat recommande de minimiser le nombre de couches. Vous pouvez atteindre le même objectif en créant une image de couche unique au moyen de la conjonction `&&` dans l'instruction RUN :

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms" && yum update -y && yum
install -y httpd
```

Le problème avec cette approche est que la lisibilité du Containerfile se détériore. Utilisez le code d'échappement `\` pour insérer des sauts de ligne et améliorer la lisibilité. Vous pouvez également mettre des lignes en retrait pour aligner les commandes : Cet exemple crée une seule couche et améliore la lisibilité.

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms" && \
yum update -y && \
yum install -y httpd
```

Les instructions RUN, COPY et ADD créent de nouvelles couches d'image, mais RUN peut être amélioré de cette façon.

Red Hat recommande d'appliquer des règles de formatage similaires à d'autres instructions acceptant plusieurs paramètres, tels que LABEL et ENV :

```
LABEL version="2.0" \
description="This is an example container image" \
creationDate="01-09-2017"
```

```
ENV MYSQL_ROOT_PASSWORD="my_password" \
MYSQL_DATABASE="my_database"
```

## Génération d'images avec Podman

La commande `podman build` traite le Containerfile et crée une nouvelle image en fonction des instructions qu'elle contient. La syntaxe de cette commande est la suivante :

```
[user@host ~]$ podman build -t NAME:TAG DIR
```

*DIR* correspond au chemin d'accès au répertoire de travail. Il peut s'agir du répertoire actuel désigné par un point (.) si le répertoire de travail est le répertoire courant. *NAME:TAG* est un nom avec une balise affectée à la nouvelle image. Si *TAG* n'est pas spécifié, l'image reçoit automatiquement la balise `latest`.



### Note

Par défaut, le répertoire de travail courant est le chemin d'accès du Containerfile, mais vous pouvez spécifier un répertoire différent à l'aide de l'indicateur `-f`. Pour plus d'informations, vous pouvez consulter Les bonnes pratiques pour écrire des Dockerfiles [[https://docs.docker.com/develop/develop-images/dockerfile\\_best-practices/](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/)].



### Références

#### Guide de référence Dockerfile

<https://docs.docker.com/engine/reference/builder/>

#### Création d'images de base

<https://docs.docker.com/engine/userguide/eng-image/baseimages/>

## ► Exercice guidé

# Création d'une image de conteneur Apache de base

Dans cet exercice, vous allez créer une image de base de conteneur Apache.

## Résultats

Vous devez pouvoir créer une image de conteneur Apache personnalisée et compilée sur une image Red Hat Universal Base Image 8

## Avant De Commencer

Exécutez la commande suivante pour télécharger les fichiers d'atelier appropriés et pour vérifier que Docker est en cours d'exécution :

```
[student@workstation ~]$ lab dockerfile-create start
```

## Instructions

### ► 1. Créez le Containerfile Apache.

- 1.1. Ouvrez un terminal sur `workstation`. À l'aide de votre éditeur préféré, créez un nouveau Containerfile.

```
[student@workstation ~]$ vim ~/D0180/labs/dockerfile-create/Containerfile
```

- 1.2. Utilisez UBI 8.5 en tant qu'image de base en ajoutant l'instruction `FROM` suivante en haut du nouveau fichier Containerfile :

```
FROM ubi8/ubi:8.5
```

- 1.3. Sous l'instruction `FROM`, incluez l'instruction `MAINTAINER` pour définir le champ `Author` dans la nouvelle image. Remplacez les valeurs par vos nom et adresse électronique.

```
MAINTAINER Your Name <youremail>
```

- 1.4. Sous l'instruction `MAINTAINER`, ajoutez l'instruction `LABEL` suivante pour inclure les métadonnées de description dans la nouvelle image :

```
LABEL description="A custom Apache container based on UBI 8"
```

- 1.5. Ajoutez une instruction `RUN` avec la commande `yum install` pour installer Apache sur le nouveau conteneur.

```
RUN yum install -y httpd && \
    yum clean all
```

- 1.6. Ajoutez une instruction RUN pour remplacer le contenu de la page d'accueil HTTPD par défaut.

```
RUN echo "Hello from Containerfile" > /var/www/html/index.html
```

- 1.7. Utilisez l'instruction EXPOSE en dessous de l'instruction RUN pour renseigner le port écouté par le conteneur lors de l'exécution. Dans cette instance, définissez le port sur 80, car il s'agit du port par défaut d'un serveur Apache.

```
EXPOSE 80
```



#### Note

L'instruction EXPOSE ne rend pas le port spécifié disponible pour l'hôte ; l'instruction sert de métadonnées indiquant les ports sur lesquels le conteneur écoute.

- 1.8. À la fin du fichier, utilisez l'instruction CMD suivante pour définir httpd comme point d'accès par défaut :

```
CMD ["httpd", "-D", "FOREGROUND"]
```

- 1.9. Vérifiez que le Containerfile correspond au suivant avant d'enregistrer et de poursuivre avec les étapes suivantes :

```
FROM ubi8/ubi:8.5

MAINTAINER Your Name <youremail>

LABEL description="A custom Apache container based on UBI 8"

RUN yum install -y httpd && \
    yum clean all

RUN echo "Hello from Containerfile" > /var/www/html/index.html

EXPOSE 80

CMD ["httpd", "-D", "FOREGROUND"]
```

- 2. Créez et vérifiez l'image de conteneur Apache.

- 2.1. Utilisez les commandes suivantes pour créer une image de conteneur Apache de base à l'aide du Containerfile récemment créé :

```
[student@workstation ~]$ cd ~/D0180/labs/dockerfile-create

[student@workstation dockerfile-create]$ podman build --layers=false \
> -t do180/apache .
STEP 1/7: FROM ubi8/ubi:8.5
...output omitted...
Trying to pull registry.access.redhat.com/ubi8/ubi:8.5... ①
...output omitted...

STEP 2/7: MAINTAINER Your Name <youremail>
STEP 3/7: LABEL description="A custom Apache container based on UBI 8"
STEP 4/7: RUN yum install -y httpd &&      yum clean all
Updating Subscription Management repositories.
Unable to read consumer identity
Subscription Manager is operating in container mode.
...output omitted...
STEP 5/7: RUN echo "Hello from Containerfile" > /var/www/html/index.html
STEP 6/7: EXPOSE 80
STEP 7/7: CMD ["httpd", "-D", "FOREGROUND"]
COMMIT do180/apache
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
--> 16c8493a19a
Successfully tagged localhost/do180/apache:latest
16c8493a19ad5162031cf17c6b2e338171f4619b6358cb367608c80791aa7718
```

- ① L'image de conteneur listée dans l'instruction FROM n'est téléchargée que si elle n'est pas déjà présente dans le stockage local.



### Note

Podman crée de nombreuses images intermédiaires anonymes au cours du processus de compilation. Elles ne sont listées que si -a est utilisé. Utilisez l'option --layers=false de la sous-commande build pour indiquer à Podman de supprimer les images intermédiaires.

- 2.2. Une fois le processus de compilation terminé, exécutez podman images pour afficher la nouvelle image dans le référentiel d'images.

```
[student@workstation dockerfile-create]$ podman images
REPOSITORY                      TAG      IMAGE ID      CREATED        SIZE
localhost/do180/apache           latest   16c8493a19ad  45 seconds ago  257 MB
registry.access.redhat.com/ubi8/ubi 8.5    202c1768b1f7  2 months ago   235 MB
```

- 3. Exécutez le conteneur Apache.

- 3.1. Utilisez la commande suivante pour exécuter un conteneur à l'aide de l'image Apache :

```
[student@workstation dockerfile-create]$ podman run --name lab-apache \
> -d -p 10080:80 do180/apache
fa1d1c450e8892ae085dd8bbf763edac92c41e6ffaa7ad6ec6388466809bb391
```

3.2. Exécutez la commande `podman ps` pour afficher le conteneur en cours d'exécution.

```
[student@workstation dockerfile-create]$ podman ps
CONTAINER ID  IMAGE                   COMMAND      CREATED
STATUS        PORTS                 NAMES
fa1d1c450e88  localhost/do180/apache:latest  httpd -D ...  10 seconds ago
Up 9 seconds ago  0.0.0.0:10080->80/tcp  lab-apache
```

3.3. Utilisez la commande `curl` pour vérifier que le serveur est en cours d'exécution.

```
[student@workstation dockerfile-create]$ curl -s 127.0.0.1:10080
Hello from Containerfile
```

## Fin

Sur `workstation`, exécutez le script `lab dockerfile-create finish` pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab dockerfile-create finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Création d'images de conteneur personnalisées

Dans cet atelier, vous allez créer un Containerfile pour générer une image de conteneur de serveur Web Apache personnalisée. L'image personnalisée sera basée sur une image UBI RHEL 8.5 et fera office de page `index.html` personnalisée.

## Résultats

Vous devez pouvoir créer un conteneur de serveur Web Apache personnalisé hébergeant des fichiers HTML statiques.

## Avant De Commencer

Ouvrez un terminal sur `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab dockerfile-review start
```

## Instructions

1. Examinez le stub Containerfile fourni dans le dossier `~/D0180/labs/dockerfile-review`. Modifiez le `Containerfile` et assurez-vous qu'il est conforme aux spécifications suivantes :
  - L'image de base est `ubi8/ubi:8.5`
  - Définit le nom d'auteur et l'identifiant d'adresse électronique souhaités avec l'instruction `MAINTAINER`
  - Définit la variable d'environnement `PORT` sur 8080
  - Installe Apache (`httpd` package).
  - Modifiez le fichier de configuration Apache `/etc/httpd/conf/httpd.conf` pour faire basculer l'écoute du port 80 par défaut sur le port 8080.
  - Remplacez la propriété des dossiers `/etc/httpd/logs` et `/run/httpd` par l'utilisateur et le groupe apache (les UID et GID sont au nombre de 48).
  - Exposez l'ensemble de valeurs dans la variable d'environnement `PORT` de sorte que les utilisateurs de conteneurs sachent accéder au serveur Web Apache.
  - Copiez le contenu du dossier `src/` du répertoire d'atelier dans le fichier `DocumentRoot` d'Apache (`/var/www/html/`) à l'intérieur du conteneur.

Le dossier `src` contient un fichier `index.html` unique qui imprime le message `Hello World!`.

- Démarrez le démon `httpd` Apache en avant-plan à l'aide de l'instruction `CMD` et de la commande suivante :

```
httpd -D FOREGROUND
```

2. Créez l'image Apache personnalisée en la nommant `do180/custom-apache`.
3. Créez un conteneur en mode détaché avec les caractéristiques suivantes :

- Nom : `containerfile`
- Container image : `do180/custom-apache`
- Port forward: du port hôte 20080 au port conteneur 8080.
- Run as a daemon : oui

Vérifiez que le conteneur est prêt et en cours d'exécution.

4. Vérifiez que le serveur renvoie la page HTML.

## Évaluation

Notez votre travail en exécutant la commande `lab dockerfile-review grade` à partir de votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab dockerfile-review grade
```

## Fin

Sur `workstation`, exécutez la commande `lab dockerfile-review finish` pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab dockerfile-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Création d'images de conteneur personnalisées

Dans cet atelier, vous allez créer un Containerfile pour générer une image de conteneur de serveur Web Apache personnalisée. L'image personnalisée sera basée sur une image UBI RHEL 8.5 et fera office de page `index.html` personnalisée.

### Résultats

Vous devez pouvoir créer un conteneur de serveur Web Apache personnalisé hébergeant des fichiers HTML statiques.

### Avant De Commencer

Ouvrez un terminal sur `workstation` en tant qu'utilisateur `student` et exécutez la commande suivante :

```
[student@workstation ~]$ lab dockerfile-review start
```

### Instructions

1. Examinez le stub Containerfile fourni dans le dossier `~/D0180/labs/dockerfile-review`. Modifiez le `Containerfile` et assurez-vous qu'il est conforme aux spécifications suivantes :
  - L'image de base est `ubi8/ubi:8.5`
  - Définit le nom d'auteur et l'identifiant d'adresse électronique souhaités avec l'instruction `MAINTAINER`
  - Définit la variable d'environnement `PORT` sur 8080
  - Installe Apache (`httpd` package).
  - Modifiez le fichier de configuration Apache `/etc/httpd/conf/httpd.conf` pour faire basculer l'écoute du port 80 par défaut sur le port 8080.
  - Remplacez la propriété des dossiers `/etc/httpd/logs` et `/run/httpd` par l'utilisateur et le groupe apache (les UID et GID sont au nombre de 48).
  - Exposez l'ensemble de valeurs dans la variable d'environnement `PORT` de sorte que les utilisateurs de conteneurs sachent accéder au serveur Web Apache.
  - Copiez le contenu du dossier `src/` du répertoire d'atelier dans le fichier `DocumentRoot` d'Apache (`/var/www/html/`) à l'intérieur du conteneur.

Le dossier `src` contient un fichier `index.html` unique qui imprime le message `Hello World!`.

**chapitre 5 |** Création d'images de conteneur personnalisées

- Démarrez le démon `httpd` Apache en avant-plan à l'aide de l'instruction `CMD` et de la commande suivante :

```
httpd -D FOREGROUND
```

- 1.1. Utilisez votre éditeur préféré pour modifier le `Containerfile` situé dans le dossier `~/D0180/labs/dockerfile-review`.

```
[student@workstation ~]$ cd ~/D0180/labs/dockerfile-review  
[student@workstation dockerfile-review]$ vim Containerfile
```

- 1.2. Définissez l'image de base du fichier `Containerfile` sur `ubi8/ubi:8.5`.

```
FROM ubi8/ubi:8.5
```

- 1.3. Définissez vos nom et adresse électronique avec l'instruction `MAINTAINER`.

```
MAINTAINER Your Name <youremail>
```

- 1.4. Créez une variable d'environnement nommée `PORT` et configurez-la sur 8080.

```
ENV PORT 8080
```

- 1.5. Installez le serveur Apache HTTPD.

```
RUN yum install -y httpd && \  
      yum clean all
```

- 1.6. Modifiez le fichier de configuration du serveur HTTP Apache pour que le port écoute soit 8080 et modifiez la propriété des dossiers de travail du serveur avec une seule instruction `RUN`.

```
RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" /etc/httpd/conf/httpd.conf && \  
      chown -R apache:apache /etc/httpd/logs/ && \  
      chown -R apache:apache /run/httpd/
```

- 1.7. Utilisez l'instruction `USER` pour exécuter le conteneur en tant qu'utilisateur `apache`. Utilisez l'instruction `EXPOSE` pour renseigner le port écoute par le conteneur lors de l'exécution. Dans cette instance, définissez le port sur la variable d'environnement `PORT` qui est le port par défaut d'un serveur Apache.

```
USER apache  
  
# Expose the custom port that you provided in the ENV var  
EXPOSE ${PORT}
```

- 1.8. Copiez tous les fichiers du dossier `src` dans Apache `DocumentRoot` sous `/var/www/html`.

```
# Copy all files under src/ folder to Apache DocumentRoot (/var/www/html)
COPY ./src/ /var/www/html/
```

- 1.9. Enfin, insérez une instruction CMD pour exécuter httpd en arrière-plan, puis enregistrez le Containerfile.

```
# Start Apache in the foreground
CMD ["httpd", "-D", "FOREGROUND"]
```

2. Créez l'image Apache personnalisée en la nommant do180/custom-apache.

- 2.1. Vérifiez le Containerfile de l'image Apache personnalisée. Ce dernier devrait ressembler à ce qui suit :

```
FROM ubi8/ubi:8.5

MAINTAINER Your Name <youremail>

ENV PORT 8080

RUN yum install -y httpd && \
    yum clean all

RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" /etc/httpd/conf/httpd.conf && \
    chown -R apache:apache /etc/httpd/logs/ && \
    chown -R apache:apache /run/httpd/

USER apache

# Expose the custom port that you provided in the ENV var
EXPOSE ${PORT}

# Copy all files under src/ folder to Apache DocumentRoot (/var/www/html)
COPY ./src/ /var/www/html/
```

```
# Start Apache in the foreground
CMD ["httpd", "-D", "FOREGROUND"]
```

- 2.2. Exécutez la commande podman build pour créer l'image Apache personnalisée et nommez-la do180/custom-apache.

```
[student@workstation dockerfile-review]$ podman build --layers=false \
> -t do180/custom-apache .
STEP 1/9: FROM ubi8/ubi:8.5
Trying to pull registry.access.redhat.com/ubi8/ubi:8.5...
...output omitted...
STEP 2/9: MAINTAINER username <username@example.com>
STEP 3/9: ENV PORT 8080
STEP 4/9: RUN yum install -y httpd && yum clean all
Updating Subscription Management repositories.
Unable to read consumer identity
Subscription Manager is operating in container mode.
```

```

...output omitted...
STEP 5/9: RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" /etc/httpd/conf/httpd.conf
  && chown -R apache:apache /etc/httpd/logs/ && chown -R apache:apache /run/httpd/
STEP 6/9: USER apache
STEP 7/9: EXPOSE ${PORT}
STEP 8/9: COPY ./src/ /var/www/html/
STEP 9/9: CMD ["httpd", "-D", "FOREGROUND"]
COMMIT do180/custom-apache
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
--> 06f94fef73b
Successfully tagged localhost/do180/custom-apache:latest
06f94fef73b0b617f6e51b34856ada9d499a45efbd738e9d0cd9586f8786e181

```

- 2.3. Exécutez la commande `podman images` pour vérifier que l'image personnalisée a été créée correctement.

```
[student@workstation dockerfile-review]$ podman images
REPOSITORY           TAG      IMAGE ID      CREATED       SIZE
localhost/do180/custom-apache    latest   06f94fef73b0  2 minutes ago  257 MB
registry.access.redhat.com/ubi8/ubi  8.5     202c1768b1f7  2 months ago  235 MB
```

3. Créez un conteneur en mode détaché avec les caractéristiques suivantes :

- Nom : `containerfile`
- Container image : `do180/custom-apache`
- Port forward: du port hôte 20080 au port conteneur 8080.
- Run as a daemon : oui

Vérifiez que le conteneur est prêt et en cours d'exécution.

- 3.1. Créez et exécutez le conteneur.

```
[student@workstation dockerfile-review]$ podman run -d \
> --name containerfile -p 20080:8080 do180/custom-apache
4a64836e7105c16a1ea4206dd1941a38f587957d7dc5fd0be5f87d11854d3146
```

- 3.2. Vérifiez que le conteneur est prêt et en cours d'exécution.

```
[student@workstation dockerfile-review]$ podman ps
CONTAINER ID  IMAGE                                     COMMAND      CREATED
STATUS        PORTS          NAMES
4a64836e7105  localhost/do180/custom-apache:latest  httpd -D ...  5 seconds ago
Up 5 seconds ago  0.0.0.0:20080->8080/tcp  containerfile
```

4. Vérifiez que le serveur renvoie la page HTML.

- 4.1. Exécutez une commande `curl` sur `127.0.0.1:20080`.

```
[student@workstation dockerfile-review]$ curl -s 127.0.0.1:20080
```

Vous devriez obtenir une sortie semblable à ceci :

```
<html>
<header><title>D0180 Hello!</title></header>
<body>
  Hello World! The containerfile-review lab works!
</body>
</html>
```

## Évaluation

Notez votre travail en exécutant la commande `lab dockerfile-review grade` à partir de votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab dockerfile-review grade
```

## Fin

Sur `workstation`, exécutez la commande `lab dockerfile-review finish` pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab dockerfile-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Un **Containerfile** contient des instructions qui spécifient comment construire une image de conteneur.
- Les images de conteneur fournies par Red Hat Container Catalog ou Quay.io constituent un bon point de départ pour créer des images personnalisées pour un langage ou une technologie spécifique.
- Le processus S2I (Source-to-Image) est une alternative aux Containerfiles. S2I implémente un processus normalisé de création d'image de conteneur pour les technologies courantes à partir du code source de l'application. Cela permet aux développeurs de se concentrer sur le développement d'applications, et non sur le développement de Containerfile.



# Déploiement d'applications en conteneur dans OpenShift

### Objectif

Déployer des applications conteneur uniques sur la plateforme OpenShift Container Platform.

### Résultats

- Décrire l'architecture de Kubernetes et de Red Hat OpenShift Container Platform.
- Créer des ressources Kubernetes standard.
- Créer une route vers un service.

### Sections

- Création de ressources Kubernetes
- Déploiement d'un serveur de base de données dans OpenShift
- Création de routes
- Exposition d'un service en tant que route
- Création d'applications avec Source-to-Image
- Création d'une application en conteneur avec Source-to-Image

### Atelier

- Déploiement d'applications en conteneur dans OpenShift

# Création de ressources Kubernetes

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure de créer des ressources Kubernetes standard :

### L'outil de ligne de commande Red Hat OpenShift Container Platform (RHOCP)

La principale méthode d'interaction avec un cluster RHOCP consiste à utiliser la commande `oc`. L'utilisation de base de la commande s'effectue via ses sous-commandes avec la syntaxe suivante :

```
[user@host ~]$ oc <command>
```

Avant d'interagir avec un cluster, la plupart des opérations requièrent que l'utilisateur se connecte. La syntaxe pour se connecter est la suivante :

```
[user@host ~]$ oc login <cluster-url>
```

### Description de la syntaxe de définition de ressources de pod

RHOCP exécute des conteneurs au sein de pods Kubernetes et, pour créer un pod à partir d'une image de conteneur, OpenShift a besoin d'une *définition de ressources de pod*. Elle peut être fournie sous la forme d'un fichier texte JSON ou YAML, ou générée à partir de valeurs par défaut par la commande `oc new-app` ou la console Web OpenShift.

Un pod est un ensemble de conteneurs et d'autres ressources. Voici un exemple de définition de pod de serveur d'applications WildFly au format YAML :

```
apiVersion: v1
kind: Pod ①
metadata:
  name: wildfly ②
  labels:
    name: wildfly ③
spec:
  containers:
    - resources:
        limits:
          cpu: 0.5
      image: do276/todojee ④
      name: wildfly
      ports:
        - containerPort: 8080 ⑤
          name: wildfly
```

```

env: ⑥
  - name: MYSQL_ENV_MYSQL_DATABASE
    value: items
  - name: MYSQL_ENV_MYSQL_USER
    value: user1
  - name: MYSQL_ENV_MYSQL_PASSWORD
    value: mypa55

```

- ① Déclare le type de ressource de pod Kubernetes.
- ② Un nom unique pour un pod dans Kubernetes qui permet aux administrateurs d'exécuter des commandes sur celui-ci.
- ③ Crée une étiquette avec une clé nommée `name` que d'autres ressources dans Kubernetes, généralement en tant que service, peuvent utiliser pour la trouver.
- ④ Définit le nom de l'image du conteneur.
- ⑤ Un attribut dépendant du conteneur identifiant le port sur le conteneur qui est exposé.
- ⑥ Définit une collection de variables d'environnement.

Certains pods peuvent nécessiter des variables d'environnement qui peuvent être lues par un conteneur. Kubernetes transforme toutes les paires `name` et `value` en variables d'environnement. Par exemple, la variable `MYSQL_ENV_MYSQL_USER` est déclarée en interne par l'exécution de Kubernetes avec une valeur `user1`, et transférée à la définition d'image de conteneur. Étant donné que le conteneur utilise le même nom de variable afin d'obtenir la connexion de l'utilisateur, la valeur est utilisée par l'instance de conteneur WildFly pour définir le nom d'utilisateur qui accède à une instance de base de données MySQL.

## Description de la syntaxe de définition de ressources de service

Kubernetes fournit un réseau virtuel qui permet aux pods de différents nœuds de calcul de se connecter. Mais Kubernetes ne fournit aucun moyen facile permettant à un pod de découvrir les adresses IP des autres pods :

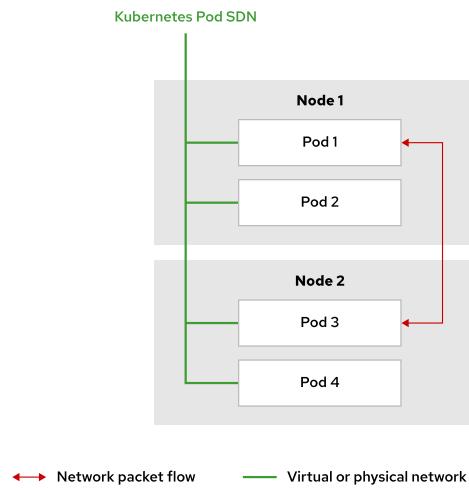


Figure 6.1: Mise en réseau Kubernetes de base

Si le pod 3 échoue et est redémarré, il peut revenir avec une adresse IP différente. Cela entraînerait l'échec du pod 1 lorsqu'il tente de communiquer avec le pod 3. Une couche de service fournit l'abstraction requise pour résoudre ce problème.

Les services sont des ressources essentielles pour n'importe quelle application OpenShift. Ils permettent aux conteneurs d'un pod d'ouvrir des connexions réseau vers les conteneurs d'un autre pod. Un pod peut être redémarré pour un grand nombre de raisons, et il reçoit une adresse IP interne différente à chaque fois. Au lieu d'un pod devant découvrir l'adresse IP d'un autre pod après chaque redémarrage, un service fournit une adresse IP stable pour les autres pods, quel que soit le nœud de calcul qui exécute le pod après chaque redémarrage :

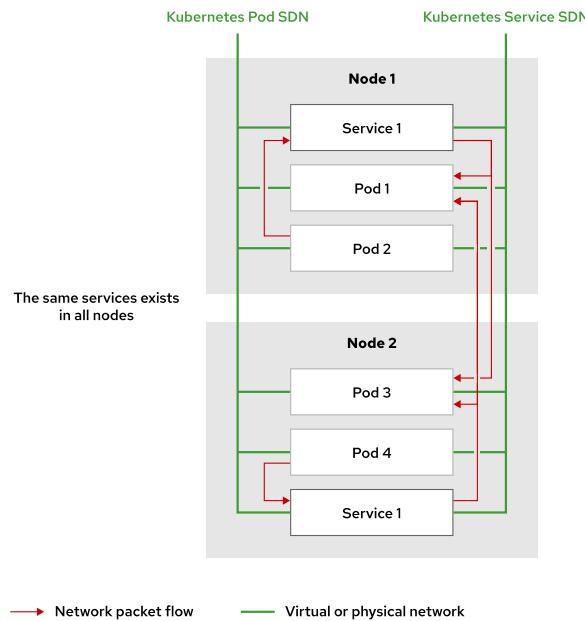


Figure 6.2: Réseau de services Kubernetes

La plupart des applications réelles ne s'exécutent pas comme un pod unique. Elles doivent effectuer une mise à l'échelle horizontale, de sorte qu'un grand nombre de pods exécutent les mêmes conteneurs à partir de la même définition de ressources de pod pour répondre à la demande croissante des utilisateurs. Un service est lié à un ensemble de pods et fournit une adresse IP unique pour l'ensemble, ainsi qu'une demande de client d'équilibrage de charge entre les pods membres.

L'ensemble des pods exécutés derrière un service sont gérés par une ressource Deployment. Une ressource Deployment intègre une ressource ReplicationController qui gère la façon dont les copies (répliques) de pods doivent être créées, et en crée de nouvelles si certaines d'entre elles échouent. Les ressources Deployment et ReplicationControllers sont expliquées ultérieurement dans ce chapitre.

L'exemple suivant montre une définition de service minimal dans la syntaxe JSON :

```
{  
    "kind": "Service", ①  
    "apiVersion": "v1",  
    "metadata": {  
        "name": "quotedb" ②  
    },
```

```
"spec": {  
    "ports": [ ❸  
        {  
            "port": 3306,  
            "targetPort": 3306  
        }  
    ],  
    "selector": {  
        "name": "mysql ldb" ❹  
    }  
}
```

- ❶** Le type de ressource Kubernetes. En l'occurrence, un service.
- ❷** Nom unique pour le service.
- ❸** `ports` correspond à un ensemble d'objets décrivant les ports réseau exposés par le service. L'attribut `targetPort` doit correspondre à un attribut `containerPort` d'une définition de conteneur de pod, et l'attribut `port` est le port qui est exposé par le service. Clients connect to the service port and the service forwards packets to the pod `targetPort`.
- ❹** `selector` définit la façon dont le service trouve les pods vers lesquels transférer les paquets. Les pods cibles doivent avoir des étiquettes correspondantes dans leurs attributs de métadonnées. Si le service trouve plusieurs pods avec des libellés correspondants, il équilibre la charge des connexions réseau entre eux.

Chaque service reçoit une adresse IP unique à laquelle les clients peuvent se connecter. Cette adresse IP provient d'un autre SDN OpenShift interne, distinct du réseau interne des pods, mais visible uniquement par les pods. Chaque pod correspondant au `selector` est ajouté à la ressource de service comme point de terminaison.

## Découverte de services

Une application trouve généralement l'adresse IP et le port d'un service en utilisant des variables d'environnement. Pour chaque service à l'intérieur du projet OpenShift, les variables d'environnement suivantes sont automatiquement définies et injectées pour tous les pods au sein du même projet :

- `SVC_NAME_SERVICE_HOST` est l'adresse IP du service.
- `SVC_NAME_SERVICE_PORT` est le port TCP du service.



### Note

La partie `SVC_NAME` de la variable est modifiée pour se conformer aux restrictions d'affectation de noms DNS : les lettres sont mises en majuscules et les traits de soulignement (`_`) sont remplacés par des tirets (`-`).

Un autre moyen de découvrir un service à partir d'un pod consiste à utiliser le serveur DNS interne d'OpenShift, qui est visible uniquement par les pods. Chaque service se voit affecter dynamiquement un enregistrement SRV avec un FQDN de la forme suivante :

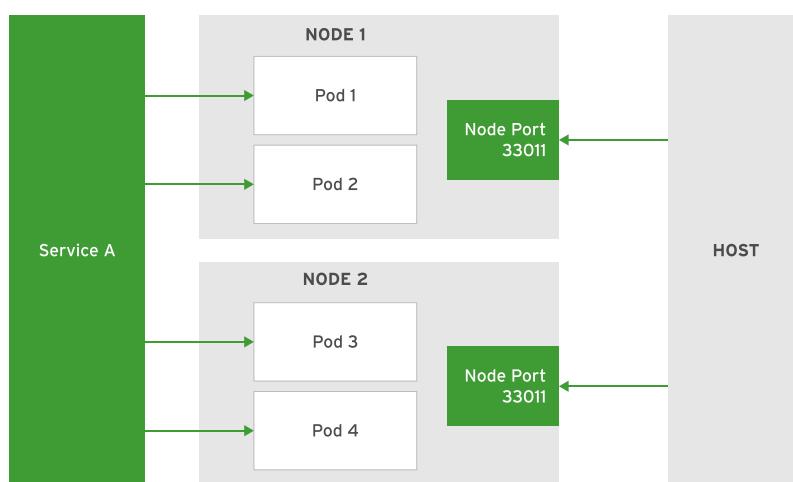
`SVC_NAME.PROJECT_NAME.svc.cluster.local`

Lors de la découverte de services à l'aide de variables d'environnement, un pod doit être créé et démarré uniquement après la création du service. Toutefois, si l'application a été écrite pour découvrir les services à l'aide de requêtes DNS, elle peut trouver les services créés après le démarrage du pod.

Une application peut accéder au service depuis l'extérieur du cluster OpenShift de deux façons différentes :

1. Type NodePort : il s'agit d'une approche plus ancienne basée sur Kubernetes, dans laquelle le service est exposé à des clients externes en s'associant aux ports disponibles sur l'hôte du nœud de calcul, qui redirige ensuite les connexions via proxy vers l'adresse IP du service. Utilisez la commande `oc edit svc` pour modifier les attributs du service et spécifiez `NodePort` comme valeur pour `type`, et fournissez une valeur de port pour l'attribut `nodePort`. OpenShift redirige alors via proxy les connexions vers le service via l'adresse IP publique de l'hôte du nœud de calcul et la valeur du port définie dans `NodePort`.
2. Routes OpenShift : approche préférée dans OpenShift pour exposer les services à l'aide d'une URL unique. Utilisez la commande `oc expose` pour exposer un service pour un accès externe ou pour exposer un service à partir de la console Web OpenShift.

*Figure 6.3 illustre comment les services NodePort permettent un accès externe aux services Kubernetes. Les routes OpenShift sont traitées plus en détail ultérieurement dans ce cours.*



**Figure 6.3: Méthode alternative pour l'accès externe à un service Kubernetes**

OpenShift fournit la commande `oc port-forward` permettant de réacheminer un port local vers un port de pod. Ce n'est pas comparable à l'accès à un pod par le biais d'une ressource de service :

- La mise en correspondance du réacheminement de ports existe uniquement sur la station de travail sur laquelle s'exécute le client `oc`, alors qu'un service fait correspondre un port pour tous les utilisateurs du réseau.
- Un service peut équilibrer la charge des connexions entre plusieurs pods, tandis que, dans le cas de la mise en correspondance du réacheminement de ports, les connexions sont réacheminées vers un seul pod.

**Note**

Red Hat déconseille l'utilisation de la méthode NodePort pour éviter l'exposition du service aux connexions directes. La mise en correspondance via le réacheminement de ports dans OpenShift est considérée comme une option plus sûre.

L'exemple suivant montre l'utilisation de la commande `oc port-forward`.

```
[user@host ~]$ oc port-forward mysql-openshift-1-glqrp 3306:3306
```

La commande réachemine le port 3306 de la machine developer vers le port 3306 sur le pod db, où un serveur MySQL (à l'intérieur d'un conteneur) accepte les connexions réseau.

**Note**

Lors de l'exécution de cette commande, veillez à ne pas fermer la fenêtre du terminal. Si vous fermez la fenêtre ou annulez le processus, la mise en correspondance du réacheminement de ports s'arrête.

## Création d'applications

Les applications simples, complexes à plusieurs niveaux et de microservice peuvent toutes être décrites par un seul fichier de définition de ressource. Ce fichier unique contient de nombreuses définitions de pods, des définitions de services pour connecter les pods, des contrôleurs de réplication ou des configurations Deployment pour permettre la mise à l'échelle horizontale des pods d'application, des demandes PersistentVolumeClaims pour assurer la persistance des données d'applications, et tous les éléments nécessaires gérables par OpenShift.

La commande `oc new-app` peut être utilisée avec l'option `-o json` ou `-o yaml` pour créer l'ossature d'un fichier de définition de ressource au format JSON ou YAML, respectivement. Ce fichier peut être personnalisé et utilisé pour créer une application à l'aide de la commande `oc create -f <filename>`, ou fusionné avec d'autres fichiers de définition de ressource pour créer une application composite.

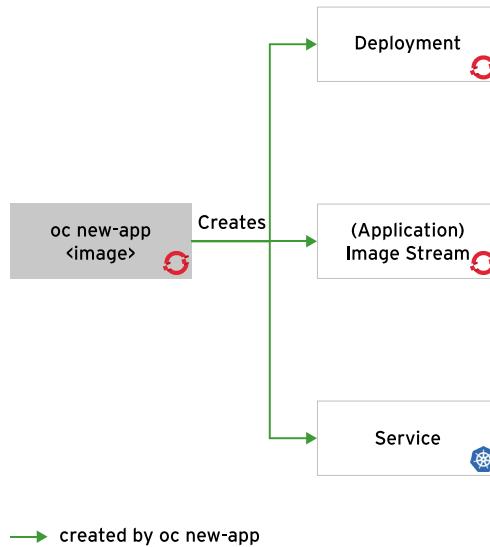
La commande `oc new-app` permet de créer des pods d'applications qui peuvent être exécutés de différentes façons dans OpenShift. Elle peut créer des pods à partir d'images Docker existantes, depuis des Dockerfiles, et à partir de code source brut à l'aide du processus S2I (Source-to-Image).

Exécutez la commande `oc new-app -h` pour comprendre les différentes options disponibles pour la création d'applications dans OpenShift.

La commande suivante crée une application sur la base d'une image, `mysql`, à partir de Docker Hub, avec le libellé défini sur `db=mysql`:

```
[user@host ~]$ oc new-app mysql MYSQL_USER=user MYSQL_PASSWORD=pass \
> MYSQL_DATABASE=testdb -l db=mysql
```

La figure suivante présente les ressources Kubernetes et OpenShift créées par la commande `oc new-app` lorsque l'argument est une image de conteneur :

**Figure 6.4: Ressources créées pour une nouvelle application**

La commande suivante crée une application basée sur une image à partir d'un registre d'image Docker privé :

```
[user@host ~]$ oc new-app --image=myregistry.com/mycompany/myapp --name=myapp
```

La commande suivante crée une application à partir d'un code source stocké dans un référentiel Git :

```
[user@host ~]$ oc new-app https://github.com/openshift/ruby-hello-world \
> --name=ruby-hello
```

Dans la section suivante, vous en apprendrez davantage sur le processus Source-to-Image (S2I), sur ses concepts associés et sur les moyens plus évolués d'utiliser oc new-app afin de compiler des applications pour OpenShift.

La commande suivante crée une application sur la base d'un modèle existant :

```
[user@host ~]$ oc new-app \
> --template=mysql-persistent \
> -p MYSQL_USER=user1 -p MYSQL_PASSWORD=mypa55 -p MYSQL_DATABASE=testdb \
> -p MYSQL_ROOT_PASSWORD=r00tpa55 -p VOLUME_CAPACITY=10Gi
...output omitted...
```

**Note**

Vous en apprendrez davantage sur les modèles dans le chapitre suivant.

## Gestion du stockage persistant

Outre la spécification des images personnalisées, vous pouvez créer un stockage persistant et l'associer à votre application. De cette façon, vous pouvez vous assurer que vos données ne sont pas perdues lors de la suppression de vos pods. Pour lister les objets PersistentVolume dans un cluster, utilisez la commande `oc get pv`:

```
[admin@host ~]$ oc get pv
NAME      CAPACITY   ACCESS MODES  RECLAIM POLICY  STATUS     CLAIM      ...
pv0001    1Mi        RWO          Retain          Available   ...
pv0002    10Mi       RWX          Recycle         Available   ...
...output omitted...
```

Pour afficher la définition YAML pour un PersistentVolume spécifique, utilisez la commande `oc get` avec l'option `-o yaml`:

```
[admin@host ~]$ oc get pv pv0001 -o yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  creationTimestamp: ...value omitted...
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    type: local
  name: pv0001
  resourceVersion: ...value omitted...
  selfLink: /api/v1/persistentvolumes/pv0001
  uid: ...value omitted...
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 1Mi
  hostPath:
    path: /data/pv0001
    type: ""
  persistentVolumeReclaimPolicy: Retain
  status:
    phase: Available
```

Pour ajouter des objets PersistentVolume à un cluster, utilisez la commande `oc create`:

```
[admin@host ~]$ oc create -f pv1001.yaml
```



### Note

Le fichier `pv1001.yaml` ci-dessus doit contenir une définition de volume persistant, de structure similaire à la sortie de la commande `oc get pv pv-name -o yaml`.

## Demande de volumes persistants

Lorsqu'une application nécessite du stockage, vous créez un objet `PersistentVolumeClaim` (PVC) pour demander une ressource de stockage dédiée à partir du pool de clusters. Le contenu suivant d'un fichier nommé `pvc.yaml` est un exemple de définition d'un PVC :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myapp
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Le PVC définit les exigences de stockage pour l'application, telles que la capacité ou le débit. Pour créer le PVC, utilisez la commande `oc create` :

```
[admin@host ~]$ oc create -f pvc.yaml
```

Après avoir créé un PVC, OpenShift tente de trouver une ressource `PersistentVolume` disponible qui répond aux exigences du PVC. Si OpenShift trouve une correspondance, il associe l'objet `PersistentVolume` à l'objet `PersistentVolumeClaim`. Pour lister les PVC dans un projet, utilisez la commande `oc get pvc` :

```
[admin@host ~]$ oc get pvc
NAME      STATUS    VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS  AGE
myapp    Bound     pv0001   1Gi       RWO          6s
```

La sortie indique si un volume persistant est lié au PVC, de pair avec les attributs du PVC (tels que la capacité).

Pour utiliser le volume persistant dans un pod d'application, définissez un montage de volume pour un conteneur qui fait référence à l'objet `PersistentVolumeClaim`. La définition du pod d'application ci-dessous fait référence à un objet `PersistentVolumeClaim` pour définir un montage de volume pour l'application :

```
apiVersion: "v1"
kind: "Pod"
metadata:
  name: "myapp"
  labels:
    name: "myapp"
spec:
  containers:
    - name: "myapp"
      image: openshift/myapp
      ports:
        - containerPort: 80
          name: "http-server"
  volumeMounts:
```

```

    - mountPath: "/var/www/html"
      name: "pvol" ❶
  volumes:
    - name: "pvol" ❷
      persistentVolumeClaim:
        claimName: "myapp" ❸

```

- ❶ Cette section déclare que le volume `pvol` est monté sur `/var/www/html` dans le système de fichiers du conteneur.
- ❷ Cette section définit le volume `pvol`.
- ❸ Le volume `pvol` fait référence au PVC `myapp`. Si OpenShift associe un volume persistant disponible au PVC `myapp`, alors le volume `pvol` fait référence à ce volume associé.

## Gestion des ressources OpenShift sur la ligne de commande

Il existe plusieurs commandes essentielles utilisées pour gérer les ressources OpenShift comme décrit ci-dessous.

Utilisez la commande `oc get` pour récupérer des informations sur les ressources du cluster. En général, cette commande affiche uniquement les principales caractéristiques des ressources et n'indique pas de détails.

La commande `oc get RESOURCE_TYPE` affiche un résumé de toutes les ressources du type spécifié. Voici un exemple de sortie de la commande `oc get pods`.

NAME	READY	STATUS	RESTARTS	AGE
nginx-1-5r583	1/1	Running	0	1h
myapp-1-l44m7	1/1	Running	0	1h

### oc get all

Utilisez la commande `oc get all` pour récupérer un résumé des composants les plus importants d'un cluster. Cette commande parcourt les principaux types de ressources pour le projet en cours et publie un récapitulatif de leurs informations :

NAME	DOCKER REPO	TAGS	UPDATED	
is/nginx	172.30.1.1:5000/basic-kubernetes/nginx	latest	About an hour ago	
<hr/>				
NAME	REVISION	DESIRED	CURRENT	
dc/nginx	1	1	1	
<hr/>				
NAME	DESIRED	CURRENT	READY	AGE
rc/nginx-1	1	1	1	1h
<hr/>				
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
svc/nginx	172.30.72.75	<none>	80/TCP, 443/TCP	1h
<hr/>				
NAME	READY	STATUS	RESTARTS	AGE
po/nginx-1-ypp8t	1/1	Running	0	1h

**oc describe**

Si les récapitulatifs obtenus via `oc get` ne suffisent pas, utilisez la commande `oc describe RESOURCE_TYPE RESOURCE_NAME` pour extraire des informations complémentaires. Contrairement à la commande `oc get`, elle ne permet pas de parcourir les différentes ressources par type. Même si la plupart des principales ressources peuvent être décrites, cette fonctionnalité n'est pas disponible pour l'ensemble des ressources. Le résultat suivant est un exemple de description d'une ressource pod :

```
Name: mysql-openshift-1-glqrp
Namespace: mysql-openshift
Priority: 0
Node: cluster-worker-1/172.25.250.52
Start Time: Fri, 15 Feb 2019 02:14:34 +0000
Labels: app=mysql-openshift
        deployment=mysql-openshift-1
...output omitted...
Status: Running
IP: 10.129.0.85
```

**oc get**

La commande `oc get RESOURCE_TYPE RESOURCE_NAME` permet d'exporter la définition d'une ressource. Elle est généralement utilisée pour créer une sauvegarde ou faciliter la modification d'une définition. Par défaut, la commande `-o yaml` publie la représentation de l'objet au format YAML, mais ce paramètre peut être modifié à l'aide de l'option `-o json`.

**oc create**

Cette commande crée des ressources à partir d'une définition de ressource. Elle s'utilise généralement avec la commande `oc get RESOURCE_TYPE RESOURCE_NAME -o yaml` pour modifier des définitions.

**oc edit**

Cette commande permet à l'utilisateur de modifier les ressources d'une définition de ressource. Par défaut, cette commande ouvre un tampon `vi` pour modifier la définition de ressource.

**oc delete**

La commande `oc delete RESOURCE_TYPE name` supprime une ressource d'un cluster OpenShift. Notez qu'une connaissance de base de l'architecture OpenShift est ici nécessaire, car la suppression de ressources gérées, telles que les pods, entraîne automatiquement la création de nouvelles instances de ces ressources. Quand un projet est supprimé, cela élimine l'intégralité des ressources et des applications qu'il contient.

**oc exec**

La commande `oc exec CONTAINER_ID options` exécute des commandes dans un conteneur. Vous pouvez l'utiliser pour exécuter des commandes par lot interactives ou non, dans le cadre d'un script.

## Ressources d'étiquetage

Lorsque vous travaillez avec plusieurs ressources dans le même projet, il est souvent utile de regrouper ces ressources par application, environnement ou d'autres critères. Pour établir ces groupes, vous définissez des étiquettes pour les ressources de votre projet. Les étiquettes font partie de la section `metadata` d'une ressource, et sont définies en tant que paires clé/valeur comme indiqué dans l'exemple suivant :

```
apiVersion: v1
kind: Service
metadata:
...contents omitted...
labels: app: nexus template: nexus-persistent-template
name: nexus
...contents omitted...
```

De nombreuses sous-commandes `oc` prennent en charge une option `-l` pour traiter les ressources d'une spécification d'étiquette. Pour la commande `oc get`, l'option `-l` faire office de sélecteur pour extraire uniquement les objets disposant d'une étiquette correspondante :

```
[user@host ~]$ oc get svc,deployments -l app=nexus
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/nexus  ClusterIP  172.30.29.218  <none>           8081/TCP    4h

NAME                           REVISION      DESIRED      CURRENT      ...
deployment.apps.openshift.io/nexus  1            1            1            ...
```



### Note

Bien que toute étiquette puisse apparaître dans les ressources, les clés `app` et `template` sont communes pour les étiquettes. Par convention, la clé `app` indique l'application liée à cette ressource. La clé `template` identifie toutes les ressources générées par le même modèle avec le nom du modèle.

Lorsque vous utilisez des modèles pour générer des ressources, les étiquettes sont particulièrement utiles. Une ressource de modèle présente une section `labels` séparée de la section `metadata.labels`. Les étiquettes définies dans la section `labels` ne s'appliquent pas au modèle lui-même, mais sont ajoutées à toutes les ressources générées par le modèle :

```
apiVersion: template.openshift.io/v1
kind: Template
labels: app: nexus template: nexus-persistent-template
metadata:
...contents omitted...
labels: maintainer: redhat
  name: nexus-persistent
...contents omitted...
objects:
- apiVersion: v1
  kind: Service
  metadata:
    name: nexus
labels: version: 1
...contents omitted...
```

L'exemple précédent définit une ressource de modèle avec une seule étiquette : `maintainer: redhat`. Le modèle génère une ressource de service avec trois étiquettes : `app: nexus`, `template: nexus-persistent-template` et `version: 1`.



## Références

Des informations supplémentaires sur les pods et services sont disponibles dans la section *Pods and Services* de la documentation OpenShift Container Platform :

### Architecture

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/architecture/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/architecture/index)

Vous trouverez des informations complémentaires sur la création d'images dans la documentation OpenShift Container Platform :

### Création d'images

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/images/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/images/index)

Les détails des étiquettes et des sélecteurs d'étiquettes sont disponibles dans la section *Working with Kubernetes Objects* pour la documentation Kubernetes :

### Étiquettes et sélecteurs

<https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/>

## ► Exercice guidé

# Déploiement d'un serveur de base de données dans OpenShift

Dans cet exercice, vous allez créer et déployer un pod de base de données MySQL dans OpenShift à l'aide de la commande `oc new-app`.

## Résultats

Vous devez pouvoir créer et déployer un pod de base de données MySQL dans OpenShift.

## Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Sur `workstation`, exécutez la commande suivante pour configurer l'environnement :

```
[student@workstation ~]$ lab openshift-resources start
```

## Instructions

- 1. Préparez l'environnement de l'atelier.

- 1.1. Chargez la configuration de votre environnement de formation.

Exécutez la commande suivante pour charger les variables d'environnement créées lors du premier exercice guidé :

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 1.2. Connectez-vous au cluster OpenShift.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 1.3. Créez un projet contenant votre nom d'utilisateur de développeur RHOCP pour les ressources que vous créez au cours de cet exercice :

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-mysql-openshift
Now using project ...output omitted...
```

- 2. Créez une application à partir du modèle `mysql-persistent` à l'aide de la commande `oc new-app`.

**chapitre 6 |** Déploiement d'applications en conteneur dans OpenShift

Cette image requiert que vous utilisiez l'option -p pour définir les variables d'environnement MYSQL\_USER, MYSQL\_PASSWORD, MYSQL\_DATABASE, MYSQL\_ROOT\_PASSWORD et VOLUME\_CAPACITY.

Utilisez l'option --template avec la commande oc new-app pour spécifier un modèle avec un stockage persistant, de sorte qu'OpenShift n'extraie pas l'image depuis Internet :

```
[student@workstation ~]$ oc new-app \
> --template=mysql-persistent \
> -p MYSQL_USER=user1 -p MYSQL_PASSWORD=mypa55 -p MYSQL_DATABASE=testdb \
> -p MYSQL_ROOT_PASSWORD=r00tpa55 -p VOLUME_CAPACITY=10Gi
--> Deploying template "openshift/mysql-persistent" to project
${RHT_OCP4_DEV_USER}-mysql-openshift
...output omitted...
--> Creating resources ...
  secret "mysql" created
  service "mysql" created
  persistentvolumeclaim "mysql" created
  deploymentconfig.apps.openshift.io "mysql" created
--> Success
  Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
  'oc expose service/mysql'
  Run 'oc status' to view your app.
```

- 3. Vérifiez que le pod MySQL a été créé avec succès et consultez les détails concernant le pod et son service.
- 3.1. Exécutez la commande oc status pour afficher l'état de la nouvelle application et pour vérifier que le déploiement de l'image MySQL a été effectué avec succès :

```
[student@workstation ~]$ oc status
In project ${RHT_OCP4_DEV_USER}-mysql-openshift on server ...

svc/mysql - 172.30.151.91:3306
...output omitted...
deployment #1 deployed 6 minutes ago - 1 pod
```

- 3.2. Listez les pods de ce projet pour vérifier que le pod MySQL est prêt à fonctionner :

```
[student@workstation ~]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
mysql-1-deploy  0/1     Completed   0          120s
mysql-1-5vfn4  1/1     Running    0          100s
```

**Note**

Remarquez le nom du pod en cours d'exécution. Vous aurez besoin de cette information pour vous connecter ultérieurement au serveur de base de données MySQL.

- 3.3. Utilisez la commande oc describe pour afficher de plus amples détails au sujet du pod :

```
[student@workstation ~]$ oc describe pod mysql-1-5vfn4
Name:           mysql-1-5vfn4
Namespace:      ${RHT_OCP4_DEV_USER}-mysql-openshift
Priority:       0
Node:          master01/192.168.50.10
Start Time:    Mon, 29 Mar 2021 16:42:13 -0400
Labels:         deployment=mysql-1
...output omitted...
Status:        Running
IP:            10.10.0.34
...output omitted...
```

La sortie de la commande `oc describe` peut contenir des erreurs liées à la sonde Readiness. Vous pouvez ignorer ces erreurs.

- 3.4. Listez les services de ce projet et vérifiez que le service permettant d'accéder au pod MySQL a été créé :

```
[student@workstation ~]$ oc get svc
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP   PORT(S)      AGE
mysql         ClusterIP  172.30.151.91  <none>        3306/TCP    10m
```

- 3.5. Extrayez les détails du service mysql à l'aide de la commande `oc describe` et remarquez que le type de service est `ClusterIP` par défaut :

```
[student@workstation ~]$ oc describe service mysql
Name:           mysql
Namespace:      ${RHT_OCP4_DEV_USER}-mysql-openshift
Labels:         app=mysql-persistent
                app.kubernetes.io/component=mysql-persistent
                app.kubernetes.io/instance=mysql-persistent
                template=mysql-persistent-template
Annotations:    openshift.io/generated-by: OpenShiftNewApp
...output omitted...
Selector:       name=mysql
Type:           ClusterIP
IP Family Policy: SingleStack
IP Families:    IPv4
IP:             172.30.151.91
IPs:            172.30.151.91
Port:           mysql  3306/TCP
TargetPort:     3306/TCP
Endpoints:     10.10.0.34:3306
Session Affinity: None
Events:         <none>
```

- 3.6. Listez les revendications de stockage persistant dans ce projet :

```
[student@workstation ~]$ oc get pvc
NAME      STATUS    VOLUME          CAPACITY  ...
STORAGECLASS
mysql     Bound     pvc-e9bf0b1f-47df-4500-afb6-77e826f76c15  10Gi     ...
           standard
```

3.7. Récupérez les détails du pvc mysql à l'aide de la commande `oc describe` :

```
[student@workstation ~]$ oc describe pvc/mysql
Name:            mysql
Namespace:       ${RHT_OCP4_DEV_USER}-mysql-openshift
StorageClass:    standard
Status:          Bound
Volume:          pvc-e9bf0b1f-47df-4500-afb6-77e826f76c15
Labels:          app=mysql-persistent
                  app.kubernetes.io/component=mysql-persistent
                  app.kubernetes.io/instance=mysql-persistent
                  template=mysql-persistent-template
Annotations:     openshift.io/generated-by: OpenShiftNewApp
...output omitted...
Capacity:        10Gi
Access Modes:    RWO
VolumeMode:      Filesystem
Used By:         mysql-1-5vfn4
...output omitted...
```

- 4. Connectez-vous au serveur de base de données MySQL et vérifiez que la base de données a été créée avec succès.

- 4.1. À partir de la machine `workstation`, configurez la redirection de port entre `workstation` et le pod de base de données s'exécutant sur OpenShift via le port 3306. Le terminal se bloque après l'exécution de la commande.

```
[student@workstation ~]$ oc port-forward mysql-1-5vfn4 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
```

- 4.2. À partir de la machine `workstation`, ouvrez un autre terminal et connectez-vous au serveur MySQL au moyen du client MySQL.

```
[student@workstation ~]$ mysql -uuser1 -pmypa55 --protocol tcp -h localhost
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.26 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql>
```

4.3. Vérifiez la création de la base de données testdb.

```
mysql> SHOW DATABASES;  
+-----+  
| Database      |  
+-----+  
| information_schema |  
| testdb          |  
+-----+  
2 rows in set (0.00 sec)
```

4.4. Quittez l'invite MySQL :

```
mysql> exit  
Bye
```

Fermez le terminal et revenez au précédent. Terminez le processus de transfert de port en appuyant sur **Ctrl+C**.

```
Forwarding from 127.0.0.1:3306 -> 3306  
Forwarding from [::1]:3306 -> 3306  
Handling connection for 3306  
^C
```

► 5. Supprimez le projet pour supprimer toutes les ressources au sein de celui-ci :

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-mysql-openshift
```

## Fin

Sur workstation, exéutez le script `lab openshift-resources finish` pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab openshift-resources finish
```

Cela met fin à cet exercice.

# Création de routes

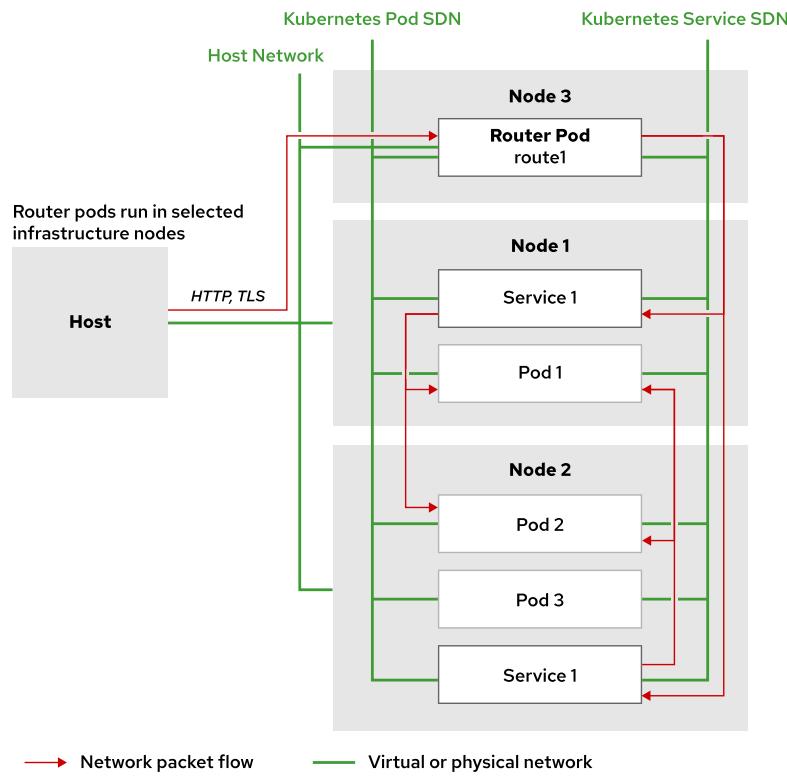
---

## Résultats

Après avoir terminé cette section, les stagiaires doivent pouvoir exposer des services au moyen de routes OpenShift.

## Travailler avec les routes

Les services permettent l'accès au réseau entre les pods situés au sein d'une instance OpenShift, et les routes permettent un accès réseau aux pods depuis les utilisateurs et les applications qui se trouvent à l'extérieur de l'instance OpenShift.



**Figure 6.5: Routes OpenShift et services Kubernetes**

Une route relie une adresse IP et un nom d'hôte DNS publics à une adresse IP de service interne. Elle utilise la ressource de service pour trouver les points de terminaison ; c'est-à-dire les ports exposés par le service.

Les routes OpenShift sont implémentées par un service de routeur à l'échelle du routeur, qui s'exécute en tant qu'application en conteneur dans le cluster OpenShift. OpenShift met à l'échelle et réplique les pods de routeur comme n'importe quelle autre application OpenShift.

**Note**

En pratique, pour améliorer les performances et réduire la latence, le routeur OpenShift se connecte directement aux pods via le réseau Software-Defined (SDN) du pod interne.

Le service de routeur utilise *HAProxy* comme implémentation par défaut.

Il est important pour les administrateurs OpenShift de noter que les noms d'hôtes DNS publics configurés pour les itinéraires doivent pointer vers les adresses IP publiques des nœuds exécutant le routeur. Les pods de routeur, contrairement aux pods d'application standard, s'associent aux adresses IP publiques de leurs nœuds, au lieu du SDN de pod interne.

L'exemple suivant montre une route minimale définie à l'aide de la syntaxe JSON :

```
{
  "apiVersion": "v1",
  "kind": "Route",
  "metadata": {
    "name": "quoteapp"
  },
  "spec": {
    "host": "quoteapp.apps.example.com",
    "to": {
      "kind": "Service",
      "name": "quoteapp"
    }
  }
}
```

Les attributs `apiVersion`, `kind` et `metadata` suivent les règles standard de définition de ressources Kubernetes. La valeur `Route` pour `kind` montre qu'il s'agit d'une ressource de route, et l'attribut `metadata.name` donne à cette route en particulier l'identifiant `quoteapp`.

Comme avec les pods et les services, la partie principale est l'attribut `spec`, qui est un objet contenant les attributs suivants :

- `host` est une chaîne contenant le FQDN associé à la route. Le DNS doit résoudre ce nom de domaine complet sur l'adresse IP du routeur OpenShift. Les informations relatives à la modification de la configuration DNS sortent du cadre de ce cours.
- `to` est un objet indiquant la ressource vers laquelle cette route pointe. Dans ce cas, la route pointe vers un service OpenShift avec le `name` défini sur `quoteapp`.

**Note**

Les noms des différents types de ressources ne sont pas en conflit. Il est permis d'avoir une route appelée `quoteapp` qui pointe vers un service également appelé `quoteapp`.

**Important**

Contrairement aux services, qui utilisent des sélecteurs pour relier des ressources de pod contenant des étiquettes spécifiques, les routes sont directement liées au nom de la ressource de service.

## Création de routes

Utilisez la commande `oc create` pour créer des ressources de route, comme toute autre ressource OpenShift. Vous devez fournir un fichier de définition de ressource JSON ou YAML, définissant la route, à la commande `oc create`.

La commande `oc new-app` ne crée pas de ressource de route lors de la création d'un pod à partir d'images de conteneur, de Containerfiles ou de code source d'application. Après tout, la commande `oc new-app` ne sait pas si le pod est destiné à être accessible en dehors de l'instance OpenShift ou non.

Un autre moyen de créer une route consiste à utiliser la commande `oc expose service`, en passant un nom de ressource de service comme entrée. L'option `--name` peut être utilisée pour contrôler le nom de la ressource de route. Par exemple :

```
[user@host ~]$ oc expose service quotedb --name quote
```

Par défaut, les routes créées par `oc expose` génèrent des noms DNS sous la forme :

```
route-name-project-name.default-domain
```

Où :

- *route-name* est le nom attribué à la route. Si aucun nom explicite n'est défini, OpenShift attribue à la route le même nom que la ressource d'origine (par exemple, le nom du service).
- *project-name* est le nom du projet contenant la ressource.
- *default-domain* est configuré sur le plan de contrôle OpenShift et correspond au domaine DNS générique répertorié comme condition préalable à l'installation d'OpenShift.

Par exemple, la création d'une route nommée `quote` dans le projet nommé `test` à partir d'une instance OpenShift dans laquelle le domaine générique est `cloudapps.example.com` a pour résultat le nom de domaine complet `quote-test.cloudapps.example.com`.

**Note**

Le serveur DNS qui héberge le domaine générique ne sait rien des noms d'hôtes de la route. Il renvoie simplement n'importe quel nom vers les adresses IP configurées. Seul le routeur OpenShift connaît les noms d'hôte de la route, traitant chacun comme un hôte virtuel HTTP. Le routeur OpenShift bloque les noms d'hôtes de domaines génériques non valides qui ne correspondent à aucune route et renvoie une erreur HTTP 404.

## Tirer parti du service de routage par défaut

Le service de routage par défaut est implémenté en tant que pod HAProxy. Les pods de routeurs, les conteneurs et leur configuration peuvent être inspectés comme toute autre ressource d'un cluster OpenShift :

```
[user@host ~]$ oc get pod --all-namespaces | grep router
openshift-ingress   router-default-746b5cfb65-f6sdm 1/1     Running   1          4d
```

Notez que vous pouvez demander des informations sur le routeur par défaut à l'aide de l'étiquette associée, comme illustré ici.

Par défaut, le routeur est déployé dans le projet `openshift-ingress`. Utilisez la commande `oc describe pod` pour obtenir les détails de la configuration de routage :

```
[user@host ~]$ oc describe pod router-default-746b5cfb65-f6sdm
Name:           router-default-746b5cfb65-f6sdm
Namespace:      openshift-ingress
...
Containers:
  router:
    ...
    ...
  Environment:
    STATS_PORT:            1936
    ROUTER_SERVICE_NAMESPACE:  openshift-ingress
    DEFAULT_CERTIFICATE_DIR: /etc/pki/tls/private
    ROUTER_SERVICE_NAME:    default
    ROUTER_CANONICAL_HOSTNAME: apps.cluster.lab.example.com
...

```

Le sous-domaine, ou le domaine par défaut à utiliser dans toutes les routes par défaut, tire sa valeur de l'entrée `ROUTER_CANONICAL_HOSTNAME`.



### Références

Des informations supplémentaires sur l'architecture des routes dans OpenShift sont disponibles dans les sections *Architecture* et *Developer Guide* de la **documentation OpenShift Container Platform**.

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/)

## ► Exercice guidé

# Exposition d'un service en tant que route

Dans cet exercice, vous allez créer, développer et déployer une application sur un cluster OpenShift et exposer son service en tant que route.

## Résultats

Vous devez savoir exposer un service en tant que route pour une application OpenShift déployée.

## Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Ouvrez un terminal sur **workstation** en tant qu'utilisateur **student** et exécutez la commande suivante :

```
[student@workstation ~]$ lab openshift-routes start
```

## Instructions

- 1. Préparez l'environnement de l'atelier.

1.1. Chargez la configuration de votre environnement de formation.

Exécutez la commande suivante pour charger les variables d'environnement créées lors du premier exercice guidé :

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

1.2. Connectez-vous au cluster OpenShift.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
```

```
Login successful
```

```
...output omitted...
```

1.3. Créez un projet contenant votre nom d'utilisateur de développeur RHOCP pour les ressources que vous créez au cours de cet exercice.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-route
```

- 2. Créez une application PHP à l'aide de l'image `quay.io/redhattraining/php-hello-dockerfile`.

- 2.1. Utilisez la commande `oc new-app` pour créer l'application PHP.



### Important

L'exemple suivant utilise la barre oblique inverse (\) pour indiquer que la deuxième ligne est la suite de la première. Si vous souhaitez ignorer la barre oblique inverse, vous pouvez taper la commande entière sur une seule ligne.

```
[student@workstation ~]$ oc new-app \
> --image=quay.io/redhattraining/php-hello-dockerfile \
> --name php-helloworld
--> Found container image 4b696cc (2 years old) from quay.io for "quay.io/
redhattraining/php-hello-dockerfile"
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose service/php-helloworld'
Run 'oc status' to view your app.
```

- 2.2. Attendez que le déploiement de l'application soit terminé en surveillant la progression à l'aide de la commande `oc get pods -w`:

```
[student@workstation ~]$ oc get pods -w
NAME                  READY   STATUS    RESTARTS   AGE
php-helloworld-74bb86f6cb-zt6wl  1/1     Running   0          5s
^C
```

Votre sortie exacte peut différer en ce qui concerne le nom, le statut, le timing et l'ordre. Le conteneur dont le statut est `Running` avec un suffixe aléatoire (`74bb86f6cb-zt6wl` dans l'exemple) contient l'application et montre qu'elle est opérationnelle.

Une autre solution consiste à surveiller les journaux de déploiement avec la commande `oc logs -f php-helloworld-74bb86f6cb-zt6wl`. Appuyez sur `Ctrl+C` pour quitter la commande si nécessaire.

```
[student@workstation ~]$ oc logs -f php-helloworld-74bb86f6cb-zt6wl
AH00558: httpd: Could not reliably determine the server's fully qualified domain
name, using 10.129.5.124. Set the 'ServerName' directive globally to suppress
this message
[09-Aug-2021 22:09:45] NOTICE: [pool www] 'user' directive is ignored when FPM is
not running as root
[09-Aug-2021 22:09:45] NOTICE: [pool www] 'group' directive is ignored when FPM is
not running as root
^C
```

Votre sortie exacte peut différer.

- 2.3. Examinez le service pour cette application à l'aide de la commande `oc describe`:

```
[student@workstation ~]$ oc describe svc/php-helloworld
Name:          php-helloworld
Namespace:    user-route
Labels:        app=php-helloworld
               app.kubernetes.io/component=php-helloworld
               app.kubernetes.io/instance=php-helloworld
Annotations:   openshift.io/generated-by: OpenShiftNewApp
Selector:      deployment=php-helloworld
Type:          ClusterIP
IP:            172.30.100.236
Port:          8080-tcp  8080/TCP
TargetPort:    8080/TCP
Endpoints:    10.129.5.124:8080
Session Affinity: None
Events:        <none>
```

L'adresse IP et l'espaces de noms affichés dans la sortie de la commande peuvent être différents.

- 3. Exposez le service, ce qui crée une route. Utilisez le nom par défaut et le nom de domaine complet (FQDN) pour la route :

```
[student@workstation ~]$ oc expose svc/php-helloworld
route.route.openshift.io/php-helloworld exposed

[student@workstation ~]$ oc describe route
Name:          php-helloworld
Namespace:    user-route
Created:      16 seconds ago
Labels:        app=php-helloworld
               app.kubernetes.io/component=php-helloworld
               app.kubernetes.io/instance=php-helloworld
Annotations:   openshift.io/host.generated=true
Requested Host:  php-helloworld-user-route.apps.na46-stage2.dev.nextcle.com
                exposed on router default
                (host apps.na46-stage2.dev.nextcle.com) 16 seconds ago
Path:          <none>
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port:  8080-tcp

Service:      php-helloworld
Weight:       100 (100%)
Endpoints:   10.129.5.124:8080
```

- ▶ 4. Accédez au service à partir d'un hôte externe au cluster pour vérifier que le service et la route fonctionnent.

```
[student@workstation ~]$ curl \
> php-helloworld-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! PHP version is 7.2.11
```



### Note

La sortie de l'application PHP dépend de la version réelle de l'image. Elle peut être différente de la vôtre.

Notez que le FQDN se compose du nom de l'application et du nom de projet par défaut. Le reste du FQDN, le sous-domaine, est défini quand OpenShift est installé.

- ▶ 5. Remplacez cette route par une route nommée xyz.

5.1. Supprimez la route actuelle :

```
[student@workstation ~]$ oc delete route/php-helloworld
route.route.openshift.io "php-helloworld" deleted
```



### Note

La suppression de la route est facultative. Vous pouvez avoir plusieurs routes pour le même service, à condition qu'elles portent des noms différents.

5.2. Créez une route pour le service avec le nom \${RHT\_OCP4\_DEV\_USER}-xyz.

```
[student@workstation ~]$ oc expose svc/php-helloworld \
> --name=${RHT_OCP4_DEV_USER}-xyz
route.route.openshift.io/${RHT_OCP4_DEV_USER}-xyz exposed

[student@workstation ~]$ oc describe route
Name:           user-xyz
Namespace:      user-route
Created:        23 seconds ago
Labels:         app=php-helloworld
                app.kubernetes.io/component=php-helloworld
                app.kubernetes.io/instance=php-helloworld
Annotations:   openshift.io/host.generated=true
Requested Host: user-xyz-user-route.apps.na46-stage2.dev.nextcle.com
                  exposed on router default
                  (host apps.na46-stage2.dev.nextcle.com) 22 seconds ago
Path:          <none>
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port:  8080-tcp
```

```
Service:      php-helloworld
Weight:      100 (100%)
Endpoints:   10.129.5.124:8080
```

Votre sortie exacte peut différer. Notez le nouveau nom de domaine complet généré en fonction du nom de la nouvelle route. Le nom de la route et le nom du projet contiennent votre nom d'utilisateur ; il apparaît donc deux fois dans le nom de domaine complet (FQDN) de la route.

- 5.3. Exécutez une requête HTTP à l'aide du nom de domaine complet sur le port 80 :

```
[student@workstation ~]$ curl \
> ${RHT_OCP4_DEV_USER}-xyz-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! PHP version is 7.2.11
```

## Fin

Sur workstation, exédez le script `lab openshift-routes finish` pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab openshift-routes finish
```

L'exercice guidé est maintenant terminé.

# Création d'applications avec Source-to-Image

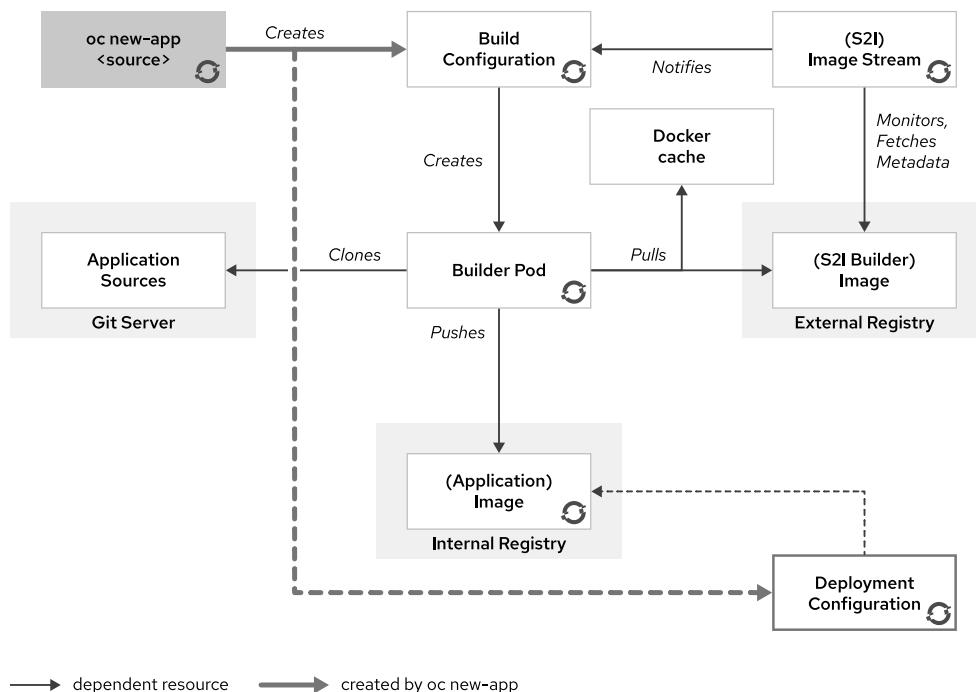
## Résultats

À la fin de cette section, les stagiaires seront en mesure de déployer une application à l'aide de la fonction Source-to-Image (S2I) d'OpenShift Container Platform.

## Le processus Source-to-Image (S2I)

Source-to-Image (S2I) est un outil qui permet de générer facilement des images de conteneur à partir du code source de l'application. Cet outil récupère le code source de l'application à partir d'un référentiel Git, injecte le code source dans un conteneur de base reposant sur le langage et le cadre souhaités, et produit une nouvelle image de conteneur qui exécute l'application assemblée.

Cette figure suivante illustre les ressources créées par la commande `oc new-app` lorsque l'argument est un référentiel de code source d'application. Notez que S2I crée également un déploiement et toutes ses ressources dépendantes :



**Figure 6.6: Déploiement et ressources dépendantes**

S2I est la stratégie principale utilisée pour le développement d'applications dans OpenShift Container Platform. Les principales raisons d'utiliser des builds sources sont les suivantes :

- Efficacité pour l'utilisateur : les développeurs n'ont pas besoin de comprendre les fichiers Dockerfiles et les commandes de système d'exploitation telles que `yum install`. Ils travaillent en utilisant leurs outils de langage de programmation standard.

- Correction de programme : S2I permet de redévelopper les applications de façon cohérente si une image de base à besoin d'un correctif du fait d'un problème de sécurité. Par exemple, si un problème de sécurité est trouvé dans une image de base PHP, la mise à jour de cette image avec des correctifs de sécurité met à jour toutes les applications qui utilisent cette image comme base.
- Rapidité : avec S2I, le processus d'assemblage peut effectuer un grand nombre d'opérations complexes sans créer de nouvelle couche à chaque étape, ce qui permet des développements plus rapides.
- Écosystème : S2I encourage un écosystème d'images partagé dans lequel les images et les scripts de base peuvent être personnalisés et réutilisés sur plusieurs types d'applications.

## Description de flux d'images

OpenShift déploie rapidement de nouvelles versions d'applications sur des pods. Pour créer une nouvelle application, une image de base (l'image S2I builder) est requise en plus du code source. Si l'un de ces deux composants est mis à jour, OpenShift crée une nouvelle image de conteneur. Les pods créés à l'aide de l'ancienne image de conteneur sont remplacés par les pods utilisant la nouvelle image.

Bien qu'il soit évident que l'image de conteneur doit être mise à jour lorsque le code d'application est modifié, il ne va pas forcément de soi que les pods déployés doivent également l'être en cas de changement d'image d'outil de compilation.

La ressource de flux d'images est une configuration qui nomme des images de conteneurs spécifiques associées aux balises de flux d'images, un alias pour ces images de conteneurs. OpenShift crée des applications en fonction d'un flux d'images. Le programme d'installation d'OpenShift remplit plusieurs flux d'images par défaut au cours de l'installation. Pour déterminer les flux d'images disponibles, utilisez la commande `oc get` de la façon suivante :

```
[user@host ~]$ oc get is -n openshift
NAME          IMAGE REPOSITORY           TAGS
cli           ...svc:5000/openshift/cli    latest
dotnet        ...svc:5000/openshift/dotnet  2.1,...,3.1-el7,latest
dotnet-runtime ...svc:5000/openshift/dotnet-runtime 2.1,...,3.1-el7,latest
httpd         ...svc:5000/openshift/httpd   2.4,2.4-el7,2.4-el8,latest
jenkins       ...svc:5000/openshift/jenkins  2,latest
mariadb       ...svc:5000/openshift/mariadb  10.3,10.3-el7,10.3-el8,latest
mongodb       ...svc:5000/openshift/mongodb  3.6,latest
mysql         ...svc:5000/openshift/mysql   8.0,8.0-el7,8.0-el8,latest
nginx         ...svc:5000/openshift/nginx   1.10,1.12,1.8,latest
nodejs        ...svc:5000/openshift/nodejs  10,...,12-ubi7,12-ubi8
perl          ...svc:5000/openshift/perl    5.26,...,5.30,5.30-el7
php           ...svc:5000/openshift/php     7.2-ubi8,...,7.3-ubi8,latest
postgresql    ...svc:5000/openshift/postgresql 10,10-el7,...,12-el7,12-el8
python        ...svc:5000/openshift/python  2.7,2.7-ubi7,...,3.6-ubi8,3.8
redis         ...svc:5000/openshift/redis   5,5-el7,5-el8,latest
ruby          ...svc:5000/openshift/ruby   2.5,2.5-ubi7,...,2.6,2.6-ubi7
...
```



### Note

Votre instance OpenShift peut compter plus ou moins de flux d'images en fonction des ajouts locaux et des versions d'OpenShift.

OpenShift détecte quand un flux d'images change et prend des mesures en fonction de ce changement. Si un problème de sécurité survient dans l'image `rhel8/nodejs-10`, cette dernière peut être mise à jour dans le référentiel d'images et OpenShift peut déclencher automatiquement une nouvelle build du code d'application.

Il est probable qu'une organisation choisisse plusieurs images S2I de base prises en charge dans Red Hat, mais elle peut également créer ses propres images de base.

## Création d'une application avec S2I et la CLI

La création d'une application avec S2I peut se faire à l'aide de l'interface de ligne de commande OpenShift.

Une application peut être créée à l'aide du processus S2I avec la commande `oc new-app` de l'interface de ligne de commande :

```
[user@host ~]$ oc new-app php~http://my.git.server.com/my-app \ ①②  
> --name=myapp ③
```

- ① Le flux d'images utilisé dans le processus s'affiche à gauche du tilde (~).
- ② L'URL après le tilde indique l'emplacement du référentiel Git du code source.
- ③ Définit le nom de l'application.



### Note

Au lieu d'utiliser le tilde, vous pouvez définir le flux d'images en utilisant l'option `-i` ou `--image-stream` pour la version complète.

```
[user@host ~]$ oc new-app -i php http://services.lab.example.com/app \  
> --name=myapp
```

La commande `oc new-app` permet la création d'applications à l'aide du code source d'un référentiel Git local ou distant. Si un seul référentiel source est spécifié, `oc new-app` essaie d'identifier le flux d'images correct à utiliser pour la création de l'application. En plus du code d'application, S2I peut également identifier et traiter les fichiers Dockerfiles pour créer une nouvelle image.

L'exemple suivant crée une application à l'aide du référentiel Git dans le répertoire actuel :

```
[user@host ~]$ oc new-app .
```



### Important

Lorsqu'un référentiel Git local est utilisé, le référentiel doit avoir une origine distante qui pointe vers une URL accessible par l'instance OpenShift.

Il est également possible de créer une application à l'aide d'un référentiel Git distant et d'un sous-répertoire de contexte :

```
[user@host ~]$ oc new-app https://github.com/openshift/sti-ruby.git \
> --context-dir=2.0/test/puma-test-app
```

Enfin, il est possible de créer une application à l'aide d'un référentiel Git distant avec une référence de branche spécifique :

```
[user@host ~]$ oc new-app https://github.com/openshift/ruby-hello-world.git#beta4
```

Si un flux d'images n'est pas spécifié dans la commande, `new-app` tente de déterminer quel outil de création de langage utiliser en fonction de la présence de certains fichiers dans la racine du référentiel :

Langue	Fichiers
Ruby	<code>Rakefile</code> , <code>Gemfile</code> , <code>config.ru</code>
Outils	<code>pom.xml</code>
Node.js	<code>app.json</code> , <code>package.json</code>
PHP	<code>index.php</code> , <code>composer.json</code>
Python	<code>requirements.txt</code> , <code>config.py</code>
Perl	<code>index.pl</code> , <code>cpanfile</code>

Une fois qu'un langage est détecté, la commande `new-app` recherche les balises de flux d'images qui le prennent en charge, ou un flux d'images qui correspond au nom de ce langage.

Créez un fichier de définition de ressources JSON à l'aide du paramètre `-o json` et de la redirection de sortie :

```
[user@host ~]$ oc -o json new-app php-http://services.lab.example.com/app \
> --name=myapp > s2i.json
```

Ce fichier de définition JSON crée une liste de ressources. La première ressource est le flux d'images :

```
...output omitted...
{
  "kind": "ImageStream", ❶
  "apiVersion": "image.openshift.io/v1",
  "metadata": {
    "name": "myapp", ❷
    "creationTimestamp": null
    "labels": {
      "app": "myapp"
    },
    "annotations": {
      "openshift.io/generated-by": "OpenShiftNewApp"
    }
  },
  "spec": {
```

```

        "lookupPolicy": {
            "local": false
        },
        "status": {
            "dockerImageRepository": ""
        }
    },
    ...output omitted...

```

- ➊ Définissez un type de ressource pour le flux d'images.
- ➋ Nommez le flux d'images myapp.

La configuration de construction (bc) est responsable de la définition des paramètres d'entrée et des déclencheurs exécutés pour transformer le code source dans une image exécutable. BuildConfig (BC) est la seconde ressource, et l'exemple suivant fournit un aperçu des paramètres utilisés par OpenShift pour créer une image exécutable.

```

...output omitted...
{
    "kind": "BuildConfig", ➊
    "apiVersion": "build.openshift.io/v1",
    "metadata": {
        "name": "myapp", ➋
        "creationTimestamp": null,
        "labels": {
            "app": "myapp"
        },
        "annotations": {
            "openshift.io/generated-by": "OpenShiftNewApp"
        }
    },
    "spec": {
        "triggers": [
            {
                "type": "GitHub",
                "github": {
                    "secret": "S5_4BZpPabM6KrIuPBvI"
                }
            },
            {
                "type": "Generic",
                "generic": {
                    "secret": "3q8K8JNDoRzhj0z1KgMz"
                }
            },
            {
                "type": "ConfigChange"
            },
            {
                "type": "ImageChange",
                "imageChange": {}
            }
        ]
    }
}

```

```

],
  "source": {
    "type": "Git",
    "git": {
      "uri": "http://services.lab.example.com/app" ③
    }
  },
  "strategy": {
    "type": "Source", ④
    "sourceStrategy": {
      "from": {
        "kind": "ImageStreamTag",
        "namespace": "openshift",
        "name": "php:7.3" ⑤
      }
    }
  },
  "output": {
    "to": {
      "kind": "ImageStreamTag",
      "name": "myapp:latest" ⑥
    }
  },
  "resources": {},
  "postCommit": {},
  "nodeSelector": null
},
"status": {
  "lastVersion": 0
}
},
...output omitted...

```

- ➊ Définissez un type de ressource `BuildConfig`.
- ➋ Nommez la ressource `BuildConfig` `myapp`.
- ➌ Définissez l'adresse du référentiel Git du code source.
- ➍ Définissez la stratégie d'utilisation de S2I.
- ➎ Définissez l'image d'outil de compilation en tant que flux d'images `php:7.3`.
- ➏ Nommez le flux d'images `myapp:latest`.

La troisième ressource est l'objet de déploiement responsable de la personnalisation du processus de déploiement dans OpenShift. Elle peut inclure des paramètres et des déclencheurs nécessaires pour créer de nouvelles instances de conteneur, et est traduite en un contrôleur de réplication de Kubernetes. Voici quelques-unes des fonctions fournies par les objets `Deployment` :

- Stratégies personnalisables par les utilisateurs pour effectuer la transition des déploiements existants vers les nouveaux.
- Définissez autant de répliques actives que souhaité et possible.
- La mise à l'échelle de la réplication dépend des tailles des anciens et des nouveaux ensembles de répliques.

```

...output omitted...
{
    "kind": "Deployment", ①
    "apiVersion": "apps/v1",
    "metadata": {
        "name": "myapp", ②
        "creationTimestamp": null,
        "labels": {
            "app": "myapp",
            "app.kubernetes.io/component": "myapp",
            "app.kubernetes.io/instance": "myapp"
        },
        "annotations": {
            "image.openshift.io/triggers": "[{"from": {"kind": "ImageStreamTag", "name": "myapp:latest"}, "path": "spec.template.spec.containers[?(@.name==\"myapp\").image]"}, {"openshift.io/generated-by": "OpenShiftNewApp"}]"
        }
    },
    "spec": {
        "replicas": 1,
        "selector": {
            "matchLabels": {
                "deployment": "myapp"
            }
        },
        "template": {
            "metadata": {
                "creationTimestamp": null,
                "labels": {
                    "deployment": "myapp" ④
                }
            },
            "annotations": {
                "openshift.io/generated-by": "OpenShiftNewApp"
            }
        },
        "spec": {
            "containers": [
                {
                    "name": "myapp", ⑤
                    "image": " ", ⑥
                    "ports": [
                        {
                            "containerPort": 8080,
                            "protocol": "TCP"
                        },
                        {
                            "containerPort": 8443,
                            "protocol": "TCP"
                        }
                    ],
                    "resources": {}
                }
            ]
        }
    }
}

```

```

        ],
    },
},
"strategy": {},
},
"status": {}
},
...output omitted...

```

- ➊ Définissez un type de ressource Deployment.
- ➋ Nommez la ressource Deployment myapp.
- ➌ Un déclencheur de changement d'image provoque la création d'un nouveau déploiement chaque fois qu'une nouvelle version de l'image myapp:latest est disponible dans le référentiel.
- ➍ Déclencheur de changement de configuration qui provoque la création d'un nouveau déploiement dès qu'un modèle de contrôleur de réPLICATION change.
- ➎ Définit l'image du conteneur à déployer : myapp:latest.
- ➏ Spécifiez les ports de conteneur. Déclencheur de changement de configuration qui provoque la création d'un nouveau déploiement dès qu'un modèle de contrôleur de réPLICATION change.

Le dernier élément est le service, déjà couvert dans les chapitres précédents :

```

...output omitted...
{
  "kind": "Service",
  "apiVersion": "v1",
  "metadata": {
    "name": "myapp",
    "creationTimestamp": null,
    "labels": {
      "app": "myapp"
      "app.kubernetes.io/component": "myapp",
      "app.kubernetes.io/instance": "myapp"
    },
    "annotations": {
      "openshift.io/generated-by": "OpenShiftNewApp"
    }
  },
  "spec": {
    "ports": [
      {
        "name": "8080-tcp",
        "protocol": "TCP",
        "port": 8080,
        "targetPort": 8080
      },
      {
        "name": "8443-tcp",
        "protocol": "TCP",
        "port": 8443,
        "targetPort": 8443
      }
    ]
  }
}

```

```
        "targetPort": 8443
    }
],
"selector": {
    "deployment": "myapp"
}
},
"status": {
    "loadBalancer": {}
}
}
```



### Note

Par défaut, la commande `oc new-app` ne crée pas de route. Vous pouvez créer une route après la création de l'application. Toutefois, elle est automatiquement créée lorsque la console Web est utilisée, car elle utilise un modèle.

Après avoir créé une nouvelle application, le processus de build démarre. Utilisez la commande `oc get builds` pour afficher une liste des versions de l'application :

```
[user@host ~]$ oc get builds
NAME          TYPE      FROM      STATUS     STARTED      DURATION
php-helloworld-1  Source   Git@9e17db8  Running   13 seconds ago
```

OpenShift permet de consulter les journaux de compilation. La commande suivante affiche les dernières lignes du journal de compilation :

```
[user@host ~]$ oc logs build/myapp-1
```



### Important

If the build is not Running yet, or OpenShift has not deployed the s2i-build pod yet, the above command throws an error. Attendez quelques instants, puis réessayez.

Déclenchez une nouvelle compilation à l'aide de la commande `oc start-build build-config-name` :

```
[user@host ~]$ oc get buildconfig
NAME          TYPE      FROM      LATEST
myapp        Source   Git       1
```

```
[user@host ~]$ oc start-build myapp
build "myapp-2" started
```

## Relation entre la compilation et le déploiement

Le pod **BuildConfig** est responsable de la création des images dans OpenShift et de leur transmission vers le registre de conteneur interne. Toute mise à jour de code source ou de contenu requiert généralement une nouvelle build pour assurer la mise à jour de l'image.

Le pod **Deployment** est responsable du déploiement des pods vers OpenShift. Le résultat de l'exécution d'un pod **Deployment** est la création de pods avec les images déployées dans le registre de conteneur interne. Tout pod existant en cours d'exécution peut être détruit, selon la manière dont la ressource **Deployment** est définie.

Les ressources **BuildConfig** et **Deployment** n'interagissent pas directement. La ressource **BuildConfig** crée ou met à jour une image de conteneur. La ressource **Deployment** réagit à la nouvelle image ou à l'image mise à jour et crée des pods à partir de l'image de conteneur.



### Références

#### **Build S2I (Source-to-Image)**

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/cicd/builds#builds-strategy-s2i-buildUnderstanding-image-builds](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/cicd/builds#builds-strategy-s2i-buildUnderstanding-image-builds)

#### **Référentiel GitHub S2I**

<https://github.com/openshift/source-to-image>

## ► Exercice guidé

# Création d'une application en conteneur avec Source-to-Image

Dans cet exercice, vous allez compiler une application à partir du code source et déployer l'application sur un cluster OpenShift.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Créer une application à partir du code source à l'aide de l'interface de ligne de commande OpenShift.
- Vérifier que le déploiement de l'application a réussi à l'aide de l'interface de ligne de commande OpenShift.

## Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Exécutez la commande suivante pour télécharger les fichiers d'atelier pertinents et configurer l'environnement :

```
[student@workstation ~]$ lab openshift-s2i start
```

## Instructions

- 1. Examinez le code source PHP de l'exemple d'application, et créez et envoyez par push une nouvelle branche nommée `s2i` à utiliser au cours de cet exercice.
- 1.1. Saisissez votre clone local du référentiel Git `D0180-apps` et extrayez la branche `master` du référentiel du cours pour vous assurer que vous démarrez cet exercice à partir d'un état correct connu :

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 1.2. Créez une nouvelle branche pour enregistrer toutes les modifications que vous avez apportées au cours de cet exercice :

```
[student@workstation D0180-apps]$ git checkout -b s2i
Switched to a new branch 's2i'

[student@workstation D0180-apps]$ git push -u origin s2i
...output omitted...
* [new branch]      s2i -> s2i
Branch 's2i' set up to track remote branch 's2i' from 'origin'.
```

- 1.3. Examinez le code source PHP de l'application, à l'intérieur du dossier `php-helloworld`.

Ouvrez le fichier `index.php` dans le dossier `~/D0180-apps/php-helloworld`:

```
<?php
print "Hello, World! php version is " . PHP_VERSION . "\n";
?>
```

L'application met en œuvre une réponse simple qui renvoie la version PHP qu'elle exécute.

► 2. Préparez l'environnement de l'atelier.

- 2.1. Chargez la configuration de votre environnement de formation.

Exécutez la commande suivante pour charger les variables d'environnement créées lors du premier exercice guidé :

```
[student@workstation D0180-apps]$ source /usr/local/etc/ocp4.config
```

- 2.2. Connectez-vous au cluster OpenShift.

```
[student@workstation D0180-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 2.3. Créez un projet contenant votre nom d'utilisateur de développeur RHOCP pour les ressources que vous créez au cours de cet exercice :

```
[student@workstation D0180-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-s2i
```

► 3. Créez une application PHP à l'aide de Source-to-Image à partir du répertoire `php-helloworld` en utilisant la branche `s2i` que vous avez créée à l'étape précédente dans votre branche du référentiel Git DO180-apps.

- 3.1. Utilisez la commande `oc new-app` pour créer l'application PHP.



### Important

L'exemple suivant utilise le signe dièse (#) pour sélectionner une branche spécifique du référentiel Git ; dans ce cas, la branche `s2i` créée à l'étape précédente.

```
[student@workstation D0180-apps]$ oc new-app php:7.3 --name=php-helloworld \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#s2i \
> --context-dir php-helloworld
```

- 3.2. Attendez que la compilation soit terminée et que l'application soit déployée. Vérifiez que le processus de compilation démarre avec la commande `oc get pods`.

```
[student@workstation openshift-s2i]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
php-helloworld-1-build   1/1     Running   0          5s
```

- 3.3. Examinez les journaux de cette build. Utilisez le nom du pod de build pour cette build, `php-helloworld-1-build`.

```
[student@workstation D0180-apps]$ oc logs --all-containers \
> -f php-helloworld-1-build
...output omitted...

Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-...
php@sha256:3206...37b4
Adding transient rw bind mount for /run/secrets/rhsm
STEP 1: FROM image-registry.openshift-image-registry.svc:5000...
...output omitted...

STEP 8: RUN /usr/libexec/s2i/assemble
...output omitted...

Pushing image .../php-helloworld:latest ...
Getting image source signatures
...output omitted...

Writing manifest to image destination
Storing signatures
Successfully pushed .../php-
helloworld@sha256:3f1cdb278548c7f24429e2469c51ae35482d54e2616d596ab6fb59d6b432c454
Push successful
Cloning "https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps" ...
Commit: 9a042f7e3650ef38ad07af83b74f57c7a7d1820c (Added start up script)
...output omitted...
```

Notez que le clone du référentiel Git est la première étape de la build. Ensuite, le processus Source-to-Image a créé une nouvelle image appelée  `${RHT_OCP4_DEV_USER}-s2i/php-helloworld:latest`. La dernière étape du processus de build consiste à transmettre cette image au registre privé d'OpenShift.

- 3.4. Examinez la ressource Deployment pour cette application :

```
[student@workstation D0180-apps]$ oc describe deployment/php-helloworld
Name:                  php-helloworld
Namespace:             ${RHT_OCP4_DEV_USER}-s2i
CreationTimestamp:     Tue, 30 Mar 2021 12:54:59 -0400
Labels:                app=php-helloworld
                      app.kubernetes.io/component=php-helloworld
                      app.kubernetes.io/instance=php-helloworld
Annotations:           deployment.kubernetes.io/revision: 2
                      image.openshift.io/triggers:
                        [{"from": {"kind": "ImageStreamTag", "name": "php-helloworld:latest"}, "fieldPath": "spec.template.spec.containers[?(.name==\"php-helloworld\")]..."}
                         openshift.io/generated-by: OpenShiftNewApp
Selector:              deployment=php-helloworld
Replicas:              1 desired | 1 updated | 1 total | 1 available | 0
                       unavailable
StrategyType:          RollingUpdate
MinReadySeconds:       0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:      deployment=php-helloworld
  Annotations: openshift.io/generated-by: OpenShiftNewApp
  Containers:
    php-helloworld:
      Ports:        8080/TCP, 8443/TCP
      Host Ports:   0/TCP, 0/TCP
      Environment: <none>
      Mounts:       <none>
      Volumes:      <none>
  Conditions:
    Type        Status  Reason
    ----        ----   -----
    Available   True    MinimumReplicasAvailable
    Progressing True    NewReplicaSetAvailable
  OldReplicaSets: <none>
  NewReplicaSet:  php-helloworld-6f5d4c47ff (1/1 replicas created)
...output omitted...
```

3.5. Ajoutez une route pour tester l'application :

```
[student@workstation D0180-apps]$ oc expose service php-helloworld \
> --name ${RHT_OCP4_DEV_USER}-helloworld
route.route.openshift.io/${RHT_OCP4_DEV_USER}-helloworld exposed
```

3.6. Recherchez l'URL associée à la nouvelle route :

```
[student@workstation D0180-apps]$ oc get route -o jsonpath='{.spec.host}{"\n"}'
${RHT_OCP4_DEV_USER}-helloworld-${RHT_OCP4_DEV_USER}-s2i.
${RHT_OCP4_WILDCARD_DOMAIN}
```

**Note**

L'URL s'affiche sur une seule ligne.

- 3.7. Testez l'application en envoyant une requête HTTP GET à l'URL que vous avez obtenue lors de l'étape précédente. Saisissez l'URL sur une seule ligne.

```
[student@workstation D0180-apps]$ curl -s \
> ${RHT_OCP4_DEV_USER}-helloworld-${RHT_OCP4_DEV_USER}-s2i.\
> ${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.29
```

Le flux d'images php : 7.3 peut contenir une version plus récente de PHP 7.3.

- 4. Explorez le démarrage de compilations d'applications en modifiant l'application dans son référentiel Git et en exécutant les commandes adéquates pour démarrer une nouvelle compilation Source-to-Image.

- 4.1. Entrez le répertoire du code source.

```
[student@workstation D0180-apps]$ cd ~/D0180-apps/php-helloworld
```

- 4.2. Modifiez le fichier `index.php` comme illustré ci-dessous :

```
<?php
print "Hello, World! php version is " . PHP_VERSION . "\n";
print "A change is a coming!\n";
?>
```

Enregistrez le fichier.

- 4.3. Validez les modifications et retransmettez le code au référentiel Git distant :

```
[student@workstation php-helloworld]$ git add .
[student@workstation php-helloworld]$ git commit -m 'Changed index page contents.'
[s2i b1324aa] changed index page contents
 1 file changed, 1 insertion(+)
[student@workstation php-helloworld]$ git push origin s2i
...output omitted...
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 417 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps
 f7cd896..b1324aa s2i -> s2i
```

- 4.4. Démarrez un nouveau processus de build Source-to-Image et attendez qu'il soit compilé et déployé :

```
[student@workstation php-helloworld]$ oc start-build php-helloworld
build.build.openshift.io/php-helloworld-2 started

[student@workstation php-helloworld]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
php-helloworld-1-build 0/1     Completed  0          5m7s
php-helloworld-2-build 0/1     Completed  0          43s
...output omitted...

[student@workstation php-helloworld]$ oc logs php-helloworld-2-build -f
...output omitted...

Successfully pushed .../php-helloworld:latest@sha256:74e757a4c0edaeda497dab7...
Push successful
```



### Note

Une fois la compilation lancée, il peut s'écouler quelques secondes avant que les journaux soient disponibles. En cas d'échec de la commande précédente, attendez un peu et réessayez.

- 4.5. Une fois la deuxième compilation terminée, utilisez la commande `oc get pods` pour vérifier que la nouvelle version de l'application est en cours d'exécution.

```
[student@workstation php-helloworld]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
php-helloworld-1-build 0/1     Completed  0          11m
php-helloworld-1-deploy 0/1     Completed  0          10m
php-helloworld-2-build 0/1     Completed  0          45s
php-helloworld-2-deploy 0/1     Completed  0          16s
php-helloworld-2-wq9wz  1/1     Running   0          13s
```

- 4.6. Vérifiez que l'application diffuse le nouveau contenu. Saisissez l'URL sur une seule ligne.

```
[student@workstation php-helloworld]$ curl -s \
> ${RHT_OCP4_DEV_USER}-helloworld-${RHT_OCP4_DEV_USER}-s2i.\
> ${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.29
A change is a coming!
```

## Fin

Sur `workstation`, exéutez le script `lab openshift-s2i finish` pour mettre fin à cet atelier.

```
[student@workstation php-helloworld]$ lab openshift-s2i finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Déploiement d'applications en conteneur dans OpenShift

### Résultats

Vous devez pouvoir créer une application OpenShift et y accéder via un navigateur web.

### Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Ouvrez un terminal sur **workstation** en tant qu'utilisateur **student** et exécutez la commande suivante :

```
[student@workstation ~]$ lab openshift-review start
```

### Instructions

1. Chargez la configuration de votre environnement de formation. Connectez-vous au cluster OpenShift et créez un projet pour cet exercice. Nommez le projet \${RHT\_OCP4\_DEV\_USER}-ocp.
2. Créez une application de conversion de température nommée temps écrite en temps à l'aide de la balise de flux d'images php php:7.3.  
Le code source se trouve dans le référentiel Git sous <https://github.com/RedHatTraining/D0180-apps/> dans le répertoire temps. Vous pouvez utiliser l'interface de ligne de commande ou la console Web OpenShift pour créer l'application.
3. Vérifiez que vous pouvez accéder à l'URL de l'application dans un navigateur Web.
  - [http://temps-\\${RHT\\_OCP4\\_DEV\\_USER}-ocp.\\${RHT\\_OCP4\\_WILDCARD\\_DOMAIN}](http://temps-${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN})

### Évaluation

Sur **workstation**, exécutez la commande **lab openshift-review grade** pour noter votre travail. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab openshift-review grade
```

### Fin

Sur **workstation**, exécutez la commande **lab openshift-review finish** pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab openshift-review finish
```

L'atelier est maintenant terminé.



## ► Solution

# Déploiement d'applications en conteneur dans OpenShift

### Résultats

Vous devez pouvoir créer une application OpenShift et y accéder via un navigateur web.

### Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Ouvrez un terminal sur **workstation** en tant qu'utilisateur **student** et exécutez la commande suivante :

```
[student@workstation ~]$ lab openshift-review start
```

### Instructions

1. Chargez la configuration de votre environnement de formation. Connectez-vous au cluster OpenShift et créez un projet pour cet exercice. Nommez le projet \${RHT\_OCP4\_DEV\_USER}-ocp.
  - 1.1. Chargez la configuration de votre environnement de formation.  
Exécutez la commande suivante pour charger les variables d'environnement créées lors du premier exercice guidé :

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 1.2. Connectez-vous au cluster OpenShift.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 1.3. Créez un projet nommé \${RHT\_OCP4\_DEV\_USER}-ocp pour les ressources que vous créez au cours de cet exercice :

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-ocp
```

2. Créez une application de conversion de température nommée temps écrite en temps à l'aide de la balise de flux d'images php php:7.3.

Le code source se trouve dans le référentiel Git sous <https://github.com/RedHatTraining/D0180-apps/> dans le répertoire temps. Vous pouvez utiliser l'interface de ligne de commande ou la console Web OpenShift pour créer l'application.

2.1. Si vous utilisez l'interface de ligne de commande, exéutez les commandes suivantes :

```
[student@workstation ~]$ oc new-app \
> php:7.3-https://github.com/RedHatTraining/D0180-apps \
> --context-dir temps --name temps
--> Found image 688c0bd (2 months old) in image stream "openshift/php" under tag
"7.3" for "php:7.3"

Apache 2.4 with PHP 7.3
-----
PHP 7.3 available as container is a base platform ...output omitted...

...output omitted...

--> Creating resources ...
imagestream.image.openshift.io "temps" created
buildconfig.build.openshift.io "temps" created
deployment.apps "temps" created
service "temps" created
--> Success
Build scheduled, use « oc logs -f bc/temps » to track its progress.
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
« oc expose svc/temps »
Run 'oc status' to view your app.
```

2.2. Surveillez l'état d'avancement de la build.

```
[student@workstation ~]$ oc logs -f bc/temps
Cloning "https://github.com/RedHatTraining/D0180-apps" ...
Commit: f7cd8963ef353d9173c3a21dccc402f3616840b (Initial commit, including all
apps previously in course)
...output omitted...
Successfully pushed image-registry.openshift-image-registry.svc:5000/
${RHT_OCP4_DEV_USER}-temps
Push successful
```

2.3. Vérifiez que l'application a bien été déployée.

```
[student@workstation ~]$ oc get pods -w
NAME        READY   STATUS    RESTARTS   AGE
temps-1-build   0/1     Completed   0          91s
temps-57d678bbdd-dlz9c   1/1     Running     0          58s
```

Appuyez sur Ctrl+C pour quitter la commande `oc get pods -w`.

2.4. Exposez le service `temps` pour créer une route externe pour l'application.

```
[student@workstation ~]$ oc expose svc/temps
route.route.openshift.io/temps exposed
```

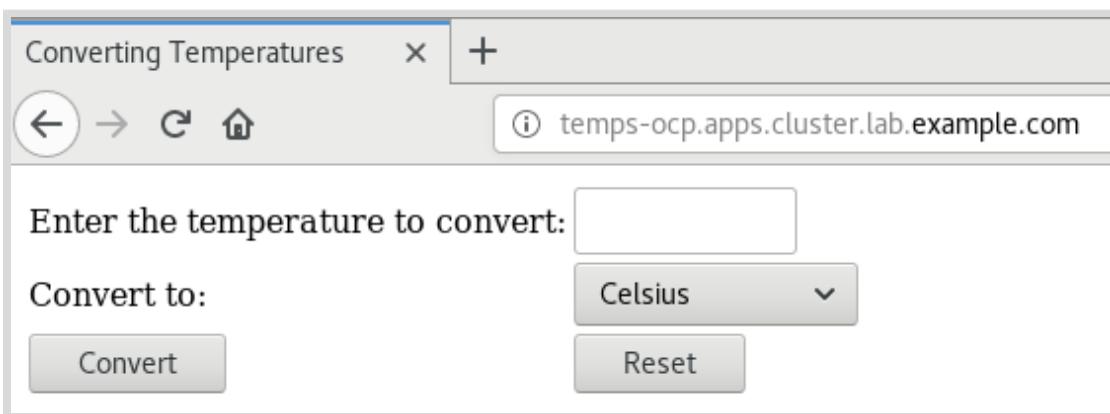
3. Vérifiez que vous pouvez accéder à l'URL de l'application dans un navigateur Web.

- `http://temps-${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN}`

3.1. Déterminez l'URL de la route.

```
[student@workstation ~]$ oc get route/temps
NAME      HOST/PORT
temps     temps-${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN} ...
```

3.2. Vérifiez que l'application de conversion de température fonctionne en ouvrant un navigateur Web et en accédant à l'URL affichée à l'étape précédente.



## Évaluation

Sur `workstation`, exécutez la commande `lab openshift-review grade` pour noter votre travail. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab openshift-review grade
```

## Fin

Sur `workstation`, exécutez la commande `lab openshift-review finish` pour mettre fin à l'atelier.

```
[student@workstation ~]$ lab openshift-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- OpenShift Container Platform stocke les définitions de chaque instance de ressource OpenShift ou Kubernetes en tant qu'objet du service de base de données distribuée du cluster, etcd. Les types de ressources courants sont les suivants : Pod, Persistent Volume (PV), Persistent Volume Claim (PVC), Service (SVC), Route, Deployment, DeploymentConfig et Build Configuration (BC).
- Utilisez le client de ligne de commande OpenShift oc pour effectuer les opérations suivantes :
  - Créer, modifier et supprimer des projets.
  - Créer des ressources d'application à l'intérieur d'un projet.
  - Supprimer, inspecter, modifier et exporter des ressources à l'intérieur d'un projet.
  - Consulter des fichiers journaux à partir de pods d'applications, de déploiements et d'opérations de compilation.
- La commande oc new-app permet de créer des pods d'applications de différentes manières : à partir d'une image de conteneur existante hébergée dans un registre d'images, depuis des Containerfiles, et à partir de code source brut à l'aide du processus S2I (Source-to-Image).
- Source-to-Image (S2I) est un outil qui permet de générer facilement une image de conteneur à partir du code source de l'application. Cet outil récupère le code source d'un référentiel Git, injecte le code source dans une image de conteneur sélectionnée basée sur le langage ou la technologie spécifiques, et produit une nouvelle image de conteneur qui exécute l'application assemblée.
- Une Route relie une adresse IP et un nom d'hôte DNS publics à une adresse IP de service interne. Alors que les services permettent l'accès au réseau entre les pods situés au sein d'une instance OpenShift, les routes permettent un accès réseau aux pods depuis les utilisateurs et les applications qui se trouvent en dehors de l'instance OpenShift.
- Vous pouvez créer, générer, déployer et contrôler des applications à l'aide de la console Web OpenShift.

## chapitre 7

# Déploiement d'applications multiconteneurs

### Objectif

Déployer des applications en conteneur à l'aide de plusieurs images de conteneur.

### Résultats

- Décrire les considérations relatives à la conteneurisation des applications avec plusieurs images de conteneur.

### Sections

- Déploiement d'une application multiconteneur sur OpenShift
- Création d'une application avec un modèle

### Atelier

- Mise en conteneur et déploiement d'une application logicielle

# Déploiement d'une application multiconteneur sur OpenShift

---

## Résultats

À la fin de cette section, les stagiaires seront en mesure d'effectuer les opérations suivantes :

- Décrire les différences entre Podman et Kubernetes.
- Déployer une application multiconteneur sur OpenShift.

## Comparaison de Docker et Kubernetes

L'utilisation de variables d'environnement vous permet de partager des informations entre des conteneurs avec Podman. Cependant, certaines limitations et un travail manuel restent nécessaires pour garantir la synchronisation de toutes les variables d'environnement, en particulier lorsque vous travaillez avec de nombreux conteneurs. Kubernetes fournit une approche permettant de résoudre ce problème en créant des services pour vos conteneurs, comme indiqué dans les chapitres précédents.

## Services dans Kubernetes

Les pods sont attachés à un espace de noms Kubernetes, ce qui représente un projet pour *OpenShift*. Lorsqu'un pod démarre, Kubernetes ajoute automatiquement un ensemble de variables d'environnement pour chaque service défini dans le même espace de noms.

Tout service défini sur Kubernetes génère des variables d'environnement pour l'adresse IP et le numéro de port du service. Kubernetes injecte automatiquement ces variables d'environnement dans les conteneurs à partir des pods figurant dans le même espace de noms. Ces variables d'environnement respectent généralement les conventions suivantes :

### Majuscules

Toutes les variables d'environnement sont définies par des noms en majuscules.

### Snake case

Toute variable d'environnement créée par un service est généralement composée de plusieurs mots séparés par un trait de soulignement (\_).

### Nom du service en premier

Le premier mot d'une variable d'environnement créée par un service est le nom du service.

### Type de protocole

La plupart des variables d'environnement réseau incluent le type de protocole (TCP ou UDP).

Ce sont les variables d'environnement générées par Kubernetes pour un service :

<SERVICE\_NAME>\_SERVICE\_HOST

Représente l'adresse IP activée par un service pour accéder à un pod.

<SERVICE\_NAME>\_SERVICE\_PORT

Représente le port sur lequel le port du serveur est listé.

<SERVICE\_NAME>\_PORT

Représente l'adresse, le port et le protocole fournis par le service pour un accès externe.

<SERVICE\_NAME>\_PORT\_<PORT\_NUMBER>\_<PROTOCOL>

Définit un alias pour le <SERVICE\_NAME>\_PORT.

<SERVICE\_NAME>\_PORT\_<PORT\_NUMBER>\_<PROTOCOL>\_PROTO

Identifie le type de protocole (TCP ou UDP).

<SERVICE\_NAME>\_PORT\_<PORT\_NUMBER>\_<PROTOCOL>\_PORT

Définit un alias pour <SERVICE\_NAME>\_SERVICE\_PORT.

<SERVICE\_NAME>\_PORT\_<PORT\_NUMBER>\_<PROTOCOL>\_ADDR

Définit un alias pour <SERVICE\_NAME>\_SERVICE\_HOST.

Par exemple, soit le service suivant :

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
  name: mysql
spec:
  ports:
    - protocol: TCP
      port: 3306
  selector:
    name: mysql
```

Les variables d'environnement suivantes sont disponibles pour chaque pod créé après le service, dans le même espace de noms :

```
MYSQL_SERVICE_HOST=10.0.0.11
MYSQL_SERVICE_PORT=3306
MYSQL_PORT=tcp://10.0.0.11:3306
MYSQL_PORT_3306_TCP=tcp://10.0.0.11:3306
MYSQL_PORT_3306_TCP_PROTO=tcp
MYSQL_PORT_3306_TCP_PORT=3306
MYSQL_PORT_3306_TCP_ADDR=10.0.0.11
```



### Note

Les autres noms de variables d'environnement <SERVICE\_NAME>\_PORT\_\* pertinents sont définis sur la base du protocole. L'adresse IP et le numéro de port sont définis dans la variable d'environnement <SERVICE\_NAME>\_PORT.

Par exemple, l'entrée MYSQL\_PORT=tcp://10.0.0.11:3306 mène à la création des variables d'environnement avec des noms tels que MYSQL\_PORT\_3306\_TCP, MYSQL\_PORT\_3306\_TCP\_PROTO, MYSQL\_PORT\_3306\_TCP\_PORT et MYSQL\_PORT\_3306\_TCP\_ADDR. Si le composant de protocole d'une variable d'environnement n'est pas défini, Kubernetes utilise le protocole TCP et attribue les noms de variable en conséquence.

## ► Exercice guidé

# Création d'une application sur OpenShift

Dans cet exercice, vous allez déployer l'application To Do List dans OpenShift Container Platform.

## Résultats

Vous devez pouvoir créer et de déployer une application dans OpenShift Container Platform.

## Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Vous devez disposer du code source de l'application To Do List et des fichiers d'atelier sur workstation. Pour télécharger les fichiers de l'atelier et vérifier le statut du cluster OpenShift, exécutez la commande suivante dans une nouvelle fenêtre de terminal.

```
[student@workstation ~]$ lab multicontainer-application start
```

## Instructions

- 1. Créez l'application To Do List à partir du fichier YAML fourni.

1.1. Connectez-vous à Openshift Container Platform.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config

[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.

...output omitted...

Using project "default".
```



### Note

Si la commande `oc login` vous invite à utiliser des connexions non sécurisées, répondez `y` (oui).

1.2. Créez dans OpenShift un nouveau projet *application* à utiliser pour cet exercice. Exécutez la commande suivante pour créer le projet *application*.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-application
Now using project ...output omitted...
```

## 1.3. Examinez le fichier YAML.

À l'aide de votre éditeur préféré, ouvrez et examinez le fichier d'application situé sur `~/D0180/labs/multicontainer-application/todo-app.yml`. Notez les ressources suivantes définies dans le `todo-app.yml` et examinez leurs configurations.

- La définition de pod `The todoapi` définit l'application Node.js.
- La définition de pod `mysql` définit la base de données MySQL.
- Le service `todoapi` fournit la connectivité au pod de l'application Node.js.
- Le service `mysql` fournit la connectivité au pod de la base de données MySQL.
- La définition de revendication de volume persistant `dbclaim` définit le volume MySQL `/var/lib/mysql/data`.

## 1.4. Créez des ressources d'application avec un fichier yaml donné.

Utilisez la commande `oc create` pour créer les ressources d'application. Dans la fenêtre de terminal, exécutez la commande suivante :

```
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-application
[student@workstation multicontainer-application]$ oc create -f todo-app.yml
pod/mysql created
pod/todoapi created
service/todoapi created
service/mysql created
persistentvolumeclaim/dbclaim created
```

1.5. Vérifiez le statut du déploiement à l'aide de la commande `oc get pods` avec l'option `-w` pour continuer à suivre le statut du pod. Attendez que les deux conteneurs soient en cours d'exécution. Le démarrage des deux pods peut prendre un peu de temps.

```
[student@workstation multicontainer-application]$ oc get pods -w
NAME      READY     STATUS      RESTARTS   AGE
todoapi   1/1      Running    0          27s
mysql     1/1      Running    0          27s
```

Appuyez sur `Ctrl+C` pour quitter la commande.

▶ 2. Connectez-vous au serveur de base de données MySQL et insérez les données dans la base de données `item`.2.1. À partir de la machine `workstation`, configurez la redirection de port entre `workstation` et le pod de base de données s'exécutant sur OpenShift via le port `3306`. Le terminal se bloque après l'exécution de la commande.

```
[student@workstation multicontainer-application]$ oc port-forward mysql 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
```

2.2. À partir de la machine `workstation`, ouvrez un autre terminal et insérez les données dans le serveur MySQL au moyen du client MySQL.

```
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-application
[student@workstation multicontainer-application]$ mysql -uuser1 \
> -h 127.0.0.1 -pmypa55 -P3306 items < db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- 2.3. Fermez le terminal et revenez au précédent. Terminez le processus de transfert de port en appuyant sur **Ctrl+C**.

```
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
Handling connection for 3306
^C
```

► 3. Exposez le service.

Afin de rendre l'application `To Do List` accessible via le routeur OpenShift et disponible en tant que FQDN public, utilisez la commande `oc expose` pour exposer le service `todoapi`.

Dans la fenêtre de terminal, exécutez la commande suivante.

```
[student@workstation multicontainer-application]$ oc expose service todoapi
route.route.openshift.io/todoapi exposed
```

► 4. Testez l'application.

- 4.1. Recherchez le FQDN de l'application en exécutant la commande `oc status`, et notez le FQDN pour l'application.

Dans la fenêtre de terminal, exécutez la commande suivante.

```
[student@workstation multicontainer-application]$ oc status | grep -o "http:.*com"
http://todoapi-${{RHT_OCP4_DEV_USER}}-application.${{RHT_OCP4_WILDCARD_DOMAIN}}
```

- 4.2. Utilisez `curl` afin de tester l'API REST pour l'application `To Do List`.

```
[student@workstation multicontainer-application]$ curl -w "\n" \
> $(oc status | grep -o "http:.*com")/todo/api/items/1
{"id":1,"description":"Pick up newspaper","done":false}
```



**Note**

Si vous utilisez l'option `-w "\n"` avec la commande `curl`, l'invite de shell s'affiche sur la ligne suivante au lieu de fusionner avec le résultat sur la même ligne.

- 4.3. Choisissez le répertoire `/home/student`.

```
[student@workstation multicontainer-application]$ cd ~
[student@workstation ~]$
```

- 4.4. Ouvrez Firefox sur `workstation` et saisissez l'URL de l'application `To Do List` dans le navigateur.

- `http://todoapi-$\{RHT_OCP4_DEV_USER\}-application.\$\{RHT_OCP4_WILDCARD_DOMAIN\}/todo/`

**Important**

La barre oblique de fin dans l'URL mentionnée ci-dessus est indispensable. Si vous ne l'incluez pas dans l'URL, vous risquez de rencontrer des problèmes avec l'application.

## To Do List Application

### To Do List

Id	Description	Done	
1	Pick up new...	false	<span style="color: red;">✖</span>
2	Buy groceries	true	<span style="color: red;">✖</span>

[First](#) [Previous](#) [1](#) [Next](#) [Last](#)

### Add Task

Description:

Add Description.

Completed:

[Clear](#) [Save](#)

Figure 7.1: Application To Do List

## Fin

Sur `workstation`, exéutez le script `lab multicontainer-application finish` pour mettre fin à cet atelier.

```
[student@workstation ~]$ lab multicontainer-application finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Déploiement d'applications multiconteneurs

### Résultats

Vous devez pouvoir créer une application OpenShift comprenant plusieurs conteneurs et y accéder via un navigateur Web.

### Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Ouvrez un terminal sur **workstation** en tant qu'utilisateur **student** et exécutez les commandes suivantes :

```
[student@workstation ~]$ lab multicontainer-review start
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-review
```

### Instructions

1. Connectez-vous au cluster OpenShift et créez un projet pour cet exercice. Nommez le projet \${RHT\_OCP4\_DEV\_USER}-deploy.
2. Compilez l'image de conteneur de base de données située dans le répertoire `images/mysql`, balisez-la avec `do180-mysql-80-rhel8` et publiez-la dans votre référentiel quay.io.
3. Compilez l'image de conteneur PHP située dans le répertoire `images/quote-php`, balisez-la avec `do180-quote-php` et publiez-la dans votre référentiel quay.io.



#### Mise en garde

Assurez-vous que les deux référentiels sont publics dans `quay.io`, de sorte qu'OpenShift puisse y récupérer les images. Reportez-vous à la section **Visibilité des référentiels** de l'annexe Quay pour obtenir des informations sur la façon de modifier la visibilité des référentiels.

4. Accédez au répertoire `~/D0180/labs/multicontainer-review` et examinez le fichier de modèle fourni `quote-php-template.json`.
5. Téléchargez le modèle d'application PHP de sorte que n'importe quel développeur ayant accès à votre projet puisse l'utiliser.
6. Traitez le modèle téléchargé, créez les ressources d'application et vérifiez leur statut.
7. Exposez le service `quote-php`. Autorisez l'accès à l'application PHP `Quote` via le routeur OpenShift et à partir d'un réseau externe.
8. Testez l'application avec `curl` et vérifiez qu'elle génère un message inspirant.

## Évaluation

Notez votre travail en exécutant la commande `lab multicontainer-review grade` à partir de votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab multicontainer-review grade
```

## Fin

Pour mettre fin à cet atelier, exécutez la commande `lab multicontainer-review finish` sur `workstation`.

```
[student@workstation ~]$ lab multicontainer-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Déploiement d'applications multiconteneurs

### Résultats

Vous devez pouvoir créer une application OpenShift comprenant plusieurs conteneurs et y accéder via un navigateur Web.

### Avant De Commencer

Assurez-vous d'avoir terminé *Exercice guidé: Configuration de l'environnement de formation* dans le *Chapitre 1* avant d'exécuter toute commande de cet exercice.

Ouvrez un terminal sur **workstation** en tant qu'utilisateur **student** et exécutez les commandes suivantes :

```
[student@workstation ~]$ lab multicontainer-review start
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-review
```

### Instructions

1. Connectez-vous au cluster OpenShift et créez un projet pour cet exercice. Nommez le projet \${RHT\_OCP4\_DEV\_USER}-deploy.
  - 1.1. À partir de la machine **workstation**, connectez-vous avec l'identité de l'utilisateur fourni lors du premier exercice.

```
[student@workstation multicontainer-review]$ source /usr/local/etc/ocp4.config
[student@workstation multicontainer-review]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
```

You don't have any projects.

You can try to create a new project, by running

```
oc new-project <projectname>
```



#### Note

Si la commande `oc login` vous invite à utiliser des connexions non sécurisées, répondez `y` (oui).

- 1.2. Créez un projet dans OpenShift nommé `deploy`, avec votre nom d'utilisateur OpenShift comme préfixe :

```
[student@workstation multicontainer-review]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-deploy
Now using project ...output omitted...
```

2. Compilez l'image de conteneur de base de données située dans le répertoire `images/mysql`, balisez-la avec `do180-mysql-80-rhel8` et publiez-la dans votre référentiel quay.io.

- 2.1. Connectez-vous à Red Hat Container Catalog à l'aide de votre compte Red Hat.

```
[student@workstation ~]$ podman login registry.redhat.io
Username: your-username
Password: your-password
Login Succeeded!
```

- 2.2. Créez l'image de base de données MySQL en utilisant le fichier `Containerfile` dans le répertoire `images/mysql`.

```
[student@workstation multicontainer-review]$ cd images/mysql

[student@workstation mysql]$ podman build -t do180-mysql-80-rhel8 .
STEP 1: FROM registry.redhat.io/rhel8/mysql-80:1
...output omitted...
STEP 4: COMMIT do180-mysql-80-rhel8
397a...5cfb
```

- 2.3. Envoyez l'image MySQL par push vers votre référentiel Quay.io.

Pour que l'image puisse être utilisée par OpenShift dans le modèle, attribuez-lui la balise `quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-80-rhel8` et envoyez-la (par push) au registre quay.io. Pour envoyer des images vers quay.io, vous devez d'abord vous connecter avec vos propres informations d'identification.

```
[student@workstation mysql]$ podman login quay.io -u ${RHT_OCP4_QUAY_USER}
Password: your-quay-password
Login Succeeded!
```

Pour baliser et déplacer l'image, exécutez les commandes suivantes dans la fenêtre de terminal.

```
[student@workstation mysql]$ podman tag do180-mysql-80-rhel8 \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-80-rhel8

[student@workstation mysql]$ podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-80-rhel8
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

Revenez au répertoire précédent.

```
[student@workstation mysql]$ cd ~/D0180/labs/multicontainer-review
```

- Compilez l'image de conteneur PHP située dans le répertoire `images/quote-php`, balisez-la avec `do180-quote-php` et publiez-la dans votre référentiel `quay.io`.

**Mise en garde**

Assurez-vous que les deux référentiels sont publics dans `quay.io`, de sorte qu'OpenShift puisse y récupérer les images. Reportez-vous à la section **Visibilité des référentiels** de l'annexe Quay pour obtenir des informations sur la façon de modifier la visibilité des référentiels.

- Créez l'image PHP en utilisant le Containerfile dans le répertoire `images/quote-php`.

```
[student@workstation multicontainer-review]$ cd images/quote-php

[student@workstation quote-php]$ podman build -t do180-quote-php .
STEP 1: FROM registry.access.redhat.com/ubi8/ubi
...output omitted...
STEP 8: COMMIT do180-quote-php
271f...525d
```

- Balisez et envoyez l'image PHP par push dans votre registre Quay.io.

Pour que l'image puisse être utilisée par OpenShift dans le modèle, attribuez-lui la balise `quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php` et envoyez-la à Quay.io.

```
[student@workstation quote-php]$ podman tag do180-quote-php \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php

[student@workstation quote-php]$ podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

- Accédez au répertoire `~/D0180/labs/multicontainer-review` et examinez le fichier de modèle fourni `quote-php-template.json`.

- Notez les définitions et la configuration des pods, des services et des revendications de volume persistant définis dans le modèle.

```
[student@workstation quote-php]$ cd ~/D0180/labs/multicontainer-review
```

- Téléchargez le modèle d'application PHP de sorte que n'importe quel développeur ayant accès à votre projet puisse l'utiliser.

- Utilisez la commande `oc create -f` pour télécharger le fichier de modèle vers le projet.

```
[student@workstation multicontainer-review]$ oc create -f quote-php-template.json
template.template.openshift.io/quote-php-persistent created
```

- Traitez le modèle téléchargé, créez les ressources d'application et vérifiez leur statut.

- 6.1. Utilisez la commande `oc process` pour traiter le fichier de modèle. Veillez à fournir le paramètre `RHT_OCP4_QUAY_USER` avec l'espace de noms `quay.io` où se trouvent les images. Utilisez la commande `pipe` pour envoyer le résultat vers la commande `oc create` afin de créer une application à partir du modèle.

```
[student@workstation multicontainer-review]$ oc process quote-php-persistent \
> -p RHT_OCP4_QUAY_USER=${RHT_OCP4_QUAY_USER} | oc create -f -
pod/mysql created
pod/quote-php created
service/quote-php created
service/mysql created
persistentvolumeclaim/dbinit created
persistentvolumeclaim/dbclaim created
```

- 6.2. Vérifiez le statut du déploiement à l'aide de la commande `oc get pods` avec l'option `-w` pour suivre le statut de déploiement. Attendez que les deux pods soient en cours d'exécution. Le démarrage des deux pods peut prendre un peu de temps.

```
[student@workstation multicontainer-review]$ oc get pods -w
NAME      READY   STATUS            RESTARTS   AGE
mysql     0/1    ContainerCreating   0          21s
quote-php 0/1    ContainerCreating   0          20s
quote-php  1/1    Running           0          35s
mysql     1/1    Running           0          49s
^C
```

Appuyez sur `Ctrl+C` pour quitter la commande.

7. Exposez le service `quote-php`.

Autorisez l'accès à l'application PHP `Quote` via le routeur OpenShift et à partir d'un réseau externe.

- 7.1. Utilisez la commande `oc expose` pour exposer le service `quote-php`.

```
[student@workstation multicontainer-review]$ oc expose svc quote-php
route.route.openshift.io/quote-php exposed
```

8. Testez l'application avec `curl` et vérifiez qu'elle génère un message inspirant.

- 8.1. Utilisez la commande `oc get route` pour trouver le nom de domaine complet où l'application est disponible. Notez le nom de domaine complet de l'application.

Dans la fenêtre de terminal, exécutez la commande suivante.

```
[student@workstation multicontainer-review]$ oc get route
NAME      HOST/PORT                               PATH  SERVICES   ...
quote-php  quote-php-your_dev_user-deploy.wildcard_domain  quote-php ...
```

- 8.2. Choisissez le répertoire `/home/student`.

```
[student@workstation multicontainer-review]$ cd ~
[student@workstation ~]$
```

- 8.3. Utilisez la commande `curl` afin de tester l'API REST pour l'application PHP `Quote`.

```
[student@workstation ~]$ curl -w "\n" \
> http://quote-php-${RHT_OCP4_DEV_USER}-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
Always remember that you are absolutely unique.
Just like everyone else.
```



### Note

Le texte affiché dans la sortie ci-dessus peut différer, mais la commande `curl` doit s'exécuter correctement.

## Évaluation

Notez votre travail en exécutant la commande `lab multicontainer-review grade` à partir de votre machine `workstation`. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab multicontainer-review grade
```

## Fin

Pour mettre fin à cet atelier, exécutez la commande `lab multicontainer-review finish` sur `workstation`.

```
[student@workstation ~]$ lab multicontainer-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Les réseaux définis par logiciel permettent la communication entre les conteneurs. Les conteneurs doivent être joints au même réseau de type Software-Defined pour communiquer.
- Les applications en conteneur ne peuvent pas se fier à des adresses IP ou à des noms d'hôtes fixes pour trouver les services.
- Podman utilise Container Network Interface (CNI) pour créer un réseau de type Software-Defined et joint tous les conteneurs sur l'hôte à ce réseau. Kubernetes et OpenShift créent un réseau de type Software-Defined entre tous les conteneurs dans un pod.
- Au sein du même projet, Kubernetes injecte un ensemble de variables pour chaque service dans tous les pods.
- Les modèles OpenShift automatisent la création d'applications qui se composent de plusieurs ressources. Les paramètres de modèle permettent d'utiliser les mêmes valeurs lors de la création de plusieurs ressources.



## chapitre 8

# Description de Red Hat OpenShift Container Platform

### Objectif

Décrire l'architecture de la solution OpenShift Container Platform.

### Résultats

- Décrire l'utilisation classique du produit et ses fonctions.
- Décrire l'architecture de la solution Red Hat OpenShift Container Platform.
- Décrire ce qu'est un opérateur de cluster et son fonctionnement, et nommer les principaux opérateurs de cluster.

### Sections

- Description des fonctions d'OpenShift Container Platform (et quiz)
- Description de l'architecture d'OpenShift (et quiz)
- Description des opérateurs de cluster (et quiz)

# Description des fonctions de la plateforme OpenShift Container Platform

---

## Résultats

À la fin de cette section, vous devez pouvoir décrire l'utilisation habituelle du produit et ses fonctions.

## Présentation d'OpenShift Container Platform

L'orchestration de conteneur est un catalyseur fondamental des initiatives de transformation numérique. Toutefois, comme les applications monolithiques migrent vers des services en conteneur, il peut être fastidieux de gérer ces applications avec l'infrastructure existante. Red Hat OpenShift Container Platform (RHOCP) aide les développeurs et services informatiques à mieux gérer les cycles de vie des applications.

La plateforme RHOCP s'appuie sur le projet Open Source Kubernetes et l'enrichit grâce à des fonctions qui offrent aux datacenters clients une plateforme de conteneur robuste, flexible et évolutive. Les développeurs peuvent ainsi exécuter des charges de travail dans un environnement à haute disponibilité.

Un orchestrateur de conteneur, tel qu'OpenShift Container Platform, gère un cluster de serveurs qui exécute plusieurs applications en conteneur. La famille de produits Red Hat OpenShift comprend un ensemble de solutions qui permettent d'améliorer la distribution d'applications métier dans divers environnements.

### Red Hat OpenShift Container Platform

Fournit un environnement Kubernetes destiné aux entreprises pour la création, le déploiement et la gestion d'applications en conteneur dans n'importe quel datacenter public ou privé, y compris les serveurs système nu. RHOCP est compatible avec plusieurs fournisseurs de cloud et de virtualisation, préservant ainsi les développeurs et les administrateurs d'applications des différences entre ces fournisseurs. C'est vous qui décidez du moment de la mise à jour vers des versions plus récentes et des composants supplémentaires à activer.

### Red Hat OpenShift Dedicated

Fournit un environnement OpenShift géré dans un cloud public, comme Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure ou IBM Cloud. Ce produit fournit toutes les fonctions offertes par RHOCP, mais Red Hat gère le cluster pour vous. Vous gardez un certain contrôle sur les décisions, telles que le moment de la mise à jour vers une version plus récente d'OpenShift ou de l'installation de services complémentaires.

### Red Hat OpenShift Online

Fournit une plateforme d'orchestration de conteneur publique hébergée qui offre une solution de développement, de création, de déploiement et d'hébergement d'applications dans un environnement cloud. La solution est partagée entre plusieurs clients et Red Hat gère le cycle de vie du cluster, qui inclut l'application de mises à jour ou l'intégration de nouvelles fonctionnalités.

### Red Hat OpenShift Kubernetes Engine

Fournit un sous-ensemble des fonctions présentes dans OpenShift Container Platform, telles que le système d'exploitation transactionnel allégé Red Hat Enterprise Linux CoreOS, le moteur CRI-O, la plateforme d'orchestration de conteneur Kubernetes et les

## chapitre 8 | Description de Red Hat OpenShift Container Platform

principaux services de cluster (console Web, mises à jour OTA, registre interne et Operator Lifecycle Manager, entre autres).

### Red Hat Code Ready Containers

Fournit une installation minimale d'OpenShift que vous pouvez exécuter sur un ordinateur portable à des fins de développement local et d'expérimentation.

Certains fournisseurs de cloud proposent également des offres basées sur RHOCP, qui permettent une intégration étroite avec d'autres services à partir de leurs plateformes. Ces offres sont prises en charge par le fournisseur de cloud en partenariat avec Red Hat. Microsoft Azure Red Hat OpenShift en est un exemple.

La figure suivante décrit les services et les fonctions d'OpenShift :

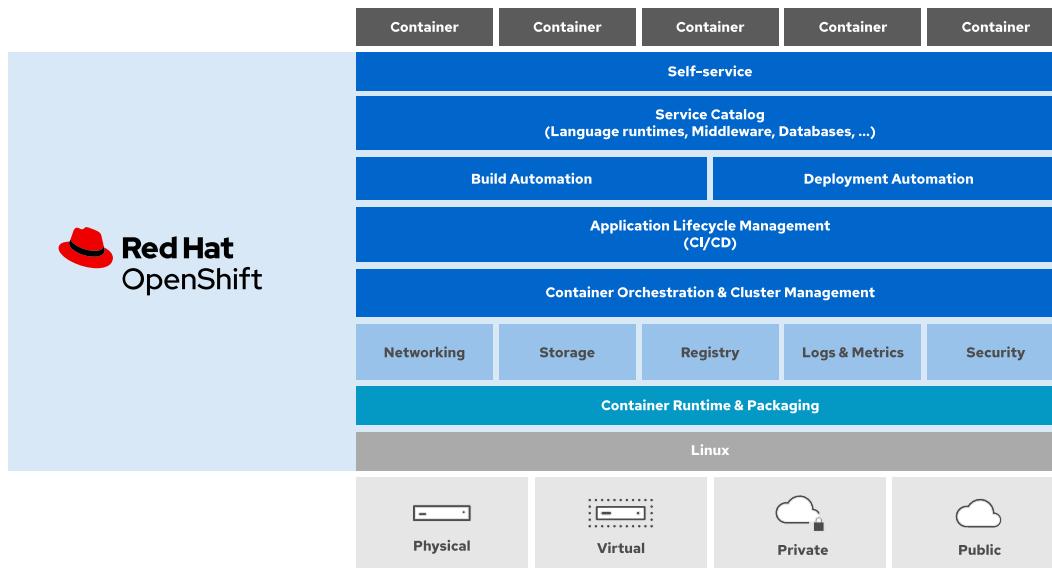


Figure 8.1: Services et fonctions d'OpenShift

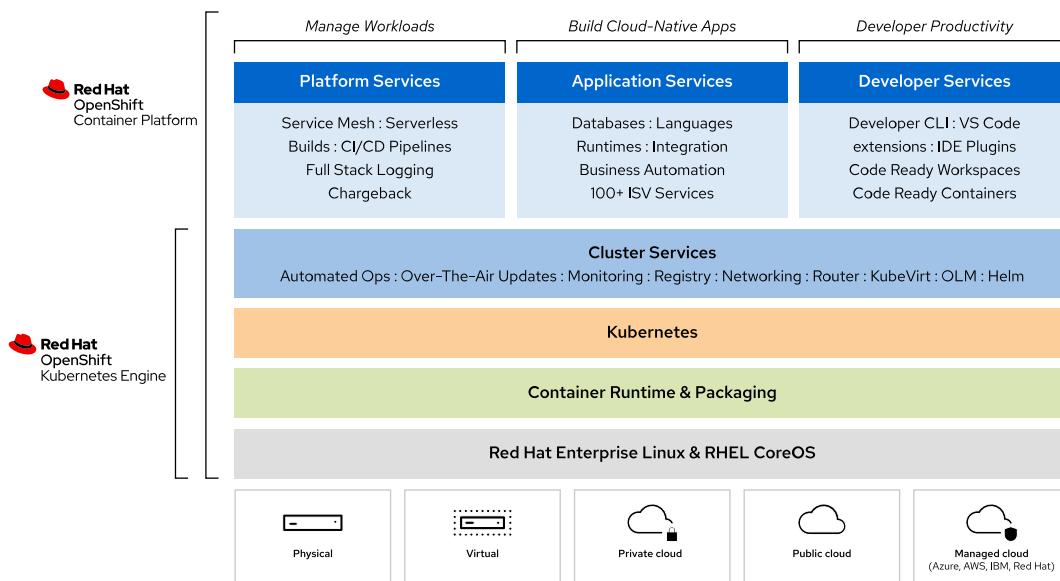
La famille de produits Red Hat OpenShift intègre de nombreux composants :

- le système d'exploitation immuable optimisé pour les conteneurs de Red Hat Enterprise Linux CoreOS ;
- le moteur CRI-O, un moteur d'exécution de conteneur compatible OCI (Open Container Initiative) à faible encombrement avec une surface d'attaque réduite ;
- Kubernetes, une plateforme d'orchestration de conteneur Open Source ;
- une console Web en libre-service ;
- un certain nombre de services d'application préinstallés, tels qu'un registre d'images de conteneur interne et une structure de surveillance ;
- des images de conteneur certifiées pour plusieurs exécutions de langage de programmation, bases de données et autres paquetages logiciels.

## Présentation des fonctions d'OpenShift

OpenShift offre de nombreuses fonctions permettant d'automatiser, de mettre à l'échelle et de gérer vos applications. Toutes ces fonctions sont activées par Kubernetes et la plupart d'entre

elles nécessitent des composants supplémentaires que vous devez ajouter et configurer sur une installation Kubernetes sur mesure.



**Figure 8.2: Comparaison des fonctions entre OpenShift Container Platform et OpenShift Kubernetes Engine**

## Haute disponibilité

Kubernetes a été conçu pour assurer une haute disponibilité, tant pour les composants internes que pour les applications utilisateur. Un cluster etcd hautement disponible stocke l'état du cluster OpenShift et de ses applications. Les ressources stockées dans etcd, telles que les configurations de déploiement, permettent le redémarrage automatique des conteneurs pour garantir que votre application est toujours en cours d'exécution et que les conteneurs défectueux sont arrêtés. Cela s'applique non seulement à vos applications, mais également aux services en conteneur qui constituent le cluster, tels que la console Web et le registre d'images interne.

## Système d'exploitation allégé

RHOCP s'exécute sur Red Hat Enterprise Linux CoreOS, le système d'exploitation allégé de Red Hat, qui privilégie l'agilité, la portabilité et la sécurité.

Red Hat Enterprise Linux CoreOS (RHEL CoreOS) est un système d'exploitation immuable, optimisé pour l'exécution d'applications en conteneur. L'intégralité du système d'exploitation est mise à jour sous la forme d'une image unique, plutôt que paquet par paquet ; les applications utilisateur et les composants système tels que les services réseau s'exécutent sous forme de conteneurs.

RHOCP contrôle les mises à jour de RHEL CoreOS et de ses configurations. Ainsi, la gestion d'un cluster OpenShift implique la gestion du système d'exploitation sur les nœuds de cluster, ce qui libère les administrateurs système de ces tâches et réduit le risque d'erreur humaine.

## Équilibrage de charge

Les clusters sont dotés de trois types de modules d'équilibrage de charge : un module d'équilibrage de charge externe qui gère l'accès à l'API OpenShift ; le module d'équilibrage de

charge HAProxy, pour les applications d'accès externe ; et le module d'équilibrage de charge interne qui utilise les règles Netfilter pour l'accès interne aux applications et aux services.

Les ressources de route utilisent HAProxy pour gérer l'accès externe au cluster. Les ressources de service utilisent des règles Netfilter pour gérer le trafic depuis l'intérieur du cluster. La technologie utilisée par les modules d'équilibrage de charge externes dépend du fournisseur de cloud qui exécute votre cluster.

## Automatisation de la mise à l'échelle

Les clusters OpenShift peuvent s'adapter à l'augmentation du trafic des applications en temps réel en démarrant automatiquement de nouveaux conteneurs et en les arrêtant lorsque la charge diminue. Cette fonction garantit que le temps d'accès de votre application reste optimal, quel que soit le nombre de connexions ou d'activités simultanées.

Les clusters OpenShift peuvent également ajouter ou supprimer davantage de nœuds de calcul du cluster en fonction de la charge agrégée à partir de nombreuses applications, ce qui permet de réduire le temps de réaction et les coûts sur les clouds publics et privés.

## Journalisation et surveillance

RHOCP est fourni avec une solution de surveillance avancée, basée sur Prometheus, qui collecte des centaines d'indicateurs sur votre cluster. Cette solution interagit avec un système d'alerte qui vous permet d'obtenir des informations détaillées sur l'activité et l'intégrité de votre cluster.

RHOCP est fourni avec une solution de journalisation agrégée avancée, basée sur Elasticsearch, qui permet de conserver à long terme des journaux à partir des nœuds de cluster et des conteneurs.

## Découverte de services

RHOCP exécute un service DNS interne sur le cluster et configure tous les conteneurs pour qu'ils utilisent ce DNS interne pour la résolution de noms. Les applications peuvent ainsi s'appuyer sur des noms conviviaux pour trouver d'autres applications et services, sans les inconvénients liés à un catalogue de services externes.

## Storage

Kubernetes ajoute une couche d'abstraction entre le backend de stockage et la consommation de stockage. Ainsi, les applications peuvent consommer du stockage à longue ou courte durée de vie, en mode bloc et basé sur des fichiers à l'aide de définitions de stockage unifiées indépendantes du backend de stockage. De cette façon, vos applications ne dépendent pas d'API de stockage de fournisseurs de cloud spécifiques.

RHOCP intègre un certain nombre de fournisseurs de stockage qui permettent le déploiement automatique du stockage sur des fournisseurs de cloud et des plateformes de virtualisation populaires ; les administrateurs de cluster n'ont donc pas besoin de gérer les petits détails des baies de stockage propriétaires.

## Gestion des applications

RHOCP permet aux développeurs d'automatiser le développement et le déploiement de leurs applications. Utilisez la fonction d'OpenShift S2I (Source-to-Image) pour créer automatiquement des conteneurs en fonction de votre code source et pour les exécuter dans OpenShift. Le registre interne stocke les images de conteneur d'application, qui peuvent être réutilisées. Cela réduit le temps nécessaire à la publication de vos applications.

Le catalogue des développeurs, accessible à partir de la console Web, est un emplacement qui permet de publier des modèles d'application et d'y accéder. Il prend en charge de nombreux langages d'exécution, tels que Python, Ruby, Java et Node.js, ainsi que des serveurs de base de données et de messagerie. Vous pouvez enrichir le catalogue en installant de nouveaux opérateurs, qui sont des applications et des services standard intégrant une intelligence opérationnelle pour le déploiement, la mise à jour et la surveillance de leurs applications.

## Extensibilité de cluster

RHOCP s'appuie sur des mécanismes d'extension standard de Kubernetes, tels que des API d'extension et des définitions de ressources personnalisées, pour ajouter des fonctions qui ne sont pas fournies par Kubernetes en amont. OpenShift fournit ces extensions en tant qu'opérateurs pour faciliter l'installation, la mise à jour et la gestion.

OpenShift comprend également OLM (Operator Lifecycle Manager), qui facilite la détection, l'installation et la mise à jour d'applications et de composants d'infrastructure fournis sous forme d'opérateurs.

Red Hat, en collaboration avec AWS, Google Cloud et Microsoft, a lancé OperatorHub, accessible à l'adresse <https://operatorhub.io/>. La plateforme est un référentiel public et une place de marché pour les opérateurs compatibles avec OpenShift et d'autres distributions de Kubernetes qui incluent OLM.

Red Hat Marketplace est une plateforme qui permet d'accéder à des logiciels certifiés empaquetés sous la forme d'opérateurs Kubernetes pouvant être déployés dans un cluster OpenShift. Les logiciels certifiés comprennent des déploiements automatiques et des mises à niveau transparentes pour garantir une expérience intégrée.



### Références

Vous trouverez des informations complémentaires dans la documentation produit de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/)

#### Red Hat OpenShift Kubernetes Engine

<https://www.openshift.com/products/kubernetes-engine>

## ► Quiz

# Description des fonctions de la plateforme OpenShift Container Platform

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les définitions suivantes, laquelle décrit le mieux les plateformes d'orchestration de conteneur ?**
  - a. Elles développent les connaissances opérationnelles de votre application et permettent de les empaqueter et de les distribuer.
  - b. Elles vous permettent de gérer un cluster de serveurs qui exécutent des applications en conteneur. Elles ajoutent des fonctionnalités telles que le libre-service, la haute disponibilité, la surveillance et l'automatisation.
  - c. Elles vous permettent de déployer des clusters IaaS sur divers fournisseurs de cloud, notamment AWS, GCP et Microsoft Azure.
  - d. Elles permettent aux développeurs d'écrire, d'empaqueter et de publier leurs applications en tant qu'opérateurs dans le catalogue d'opérateurs.
- ▶ 2. **Parmi les fonctions clés suivantes, lesquelles permettent d'assurer une haute disponibilité pour vos applications ? (Choisissez trois réponses.)**
  - a. Un cluster OpenShift etcd maintient l'état du cluster disponible pour tous les nœuds.
  - b. Les modules d'équilibrage de charge OpenShift HAProxy permettent un accès externe aux applications.
  - c. Les services OpenShift équilibrivent la charge de l'accès aux applications depuis l'intérieur du cluster.
  - d. Les configurations de déploiement OpenShift garantissent le redémarrage des conteneurs d'applications dans des scénarios tels que la perte d'un nœud.
- ▶ 3. **Parmi les propositions suivantes, lesquelles s'appliquent à OpenShift ? (Choisissez deux réponses.)**
  - a. Les développeurs peuvent créer et démarrer des applications de cloud directement depuis un référentiel de code source.
  - b. OpenShift corrige Kubernetes pour ajouter des fonctions qui ne seraient pas disponibles pour les autres distributions de Kubernetes.
  - c. OpenShift Dedicated vous donne accès à un ensemble exclusif d'opérateurs dont Red Hat assure la maintenance et la gestion. Cela permet de garantir que les opérateurs peuvent fonctionner en toute sécurité dans votre environnement.
  - d. Les administrateurs de cluster OpenShift peuvent découvrir et installer de nouveaux opérateurs à partir du catalogue d'opérateurs.

- 4. Parmi les services suivants, lesquels sont utilisés par les composants OpenShift pour équilibrer la charge de leur trafic ? (Choisissez deux réponses.)
- a. L'API OpenShift, accessible via le module d'équilibrage de charge externe.
  - b. Les services, qui utilisent Netfilter pour l'équilibrage de charge.
  - c. Les services, qui utilisent HAProxy pour l'équilibrage de charge.
  - d. Les itinéraires, qui utilisent Netfilter pour l'équilibrage de charge.
  - e. Les itinéraires, qui utilisent HAProxy pour l'équilibrage de charge.
- 5. Parmi les propositions suivantes, lesquelles s'appliquent à la haute disponibilité et à la mise à l'échelle d'OpenShift ? (Choisissez deux réponses.)
- a. OpenShift ne fournit pas la fonction de haute disponibilité par défaut. Vous devez utiliser des produits tiers à haute disponibilité.
  - b. OpenShift utilise les métriques de Prometheus pour mettre les pods d'application à l'échelle de manière dynamique.
  - c. La haute disponibilité et la mise à l'échelle sont limitées aux applications qui exposent une API REST.
  - d. OpenShift peut augmenter ou diminuer la mise à l'échelle des applications à la demande.

## ► Solution

# Description des fonctions de la plateforme OpenShift Container Platform

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les définitions suivantes, laquelle décrit le mieux les plateformes d'orchestration de conteneur ?**
  - a. Elles développent les connaissances opérationnelles de votre application et permettent de les empaqueter et de les distribuer.
  - b. Elles vous permettent de gérer un cluster de serveurs qui exécutent des applications en conteneur. Elles ajoutent des fonctionnalités telles que le libre-service, la haute disponibilité, la surveillance et l'automatisation.
  - c. Elles vous permettent de déployer des clusters IaaS sur divers fournisseurs de cloud, notamment AWS, GCP et Microsoft Azure.
  - d. Elles permettent aux développeurs d'écrire, d'empaqueter et de publier leurs applications en tant qu'opérateurs dans le catalogue d'opérateurs.
- ▶ 2. **Parmi les fonctions clés suivantes, lesquelles permettent d'assurer une haute disponibilité pour vos applications ? (Choisissez trois réponses.)**
  - a. Un cluster OpenShift etcd maintient l'état du cluster disponible pour tous les nœuds.
  - b. Les modules d'équilibrage de charge OpenShift HAProxy permettent un accès externe aux applications.
  - c. Les services OpenShift équilibrivent la charge de l'accès aux applications depuis l'intérieur du cluster.
  - d. Les configurations de déploiement OpenShift garantissent le redémarrage des conteneurs d'applications dans des scénarios tels que la perte d'un nœud.
- ▶ 3. **Parmi les propositions suivantes, lesquelles s'appliquent à OpenShift ? (Choisissez deux réponses.)**
  - a. Les développeurs peuvent créer et démarrer des applications de cloud directement depuis un référentiel de code source.
  - b. OpenShift corrige Kubernetes pour ajouter des fonctions qui ne seraient pas disponibles pour les autres distributions de Kubernetes.
  - c. OpenShift Dedicated vous donne accès à un ensemble exclusif d'opérateurs dont Red Hat assure la maintenance et la gestion. Cela permet de garantir que les opérateurs peuvent fonctionner en toute sécurité dans votre environnement.
  - d. Les administrateurs de cluster OpenShift peuvent découvrir et installer de nouveaux opérateurs à partir du catalogue d'opérateurs.

- 4. Parmi les services suivants, lesquels sont utilisés par les composants OpenShift pour équilibrer la charge de leur trafic ? (Choisissez deux réponses.)
- a. L'API OpenShift, accessible via le module d'équilibrage de charge externe.
  - b. Les services, qui utilisent Netfilter pour l'équilibrage de charge.
  - c. Les services, qui utilisent HAProxy pour l'équilibrage de charge.
  - d. Les itinéraires, qui utilisent Netfilter pour l'équilibrage de charge.
  - e. Les itinéraires, qui utilisent HAProxy pour l'équilibrage de charge.
- 5. Parmi les propositions suivantes, lesquelles s'appliquent à la haute disponibilité et à la mise à l'échelle d'OpenShift ? (Choisissez deux réponses.)
- a. OpenShift ne fournit pas la fonction de haute disponibilité par défaut. Vous devez utiliser des produits tiers à haute disponibilité.
  - b. OpenShift utilise les métriques de Prometheus pour mettre les pods d'application à l'échelle de manière dynamique.
  - c. La haute disponibilité et la mise à l'échelle sont limitées aux applications qui exposent une API REST.
  - d. OpenShift peut augmenter ou diminuer la mise à l'échelle des applications à la demande.

# Description de l'architecture d'OpenShift

## Résultats

À la fin de cette section, vous devez pouvoir décrire l'architecture de Red Hat OpenShift Container Platform.

## Présentation de l'architecture déclarative de Kubernetes

L'architecture d'OpenShift est basée sur la nature déclarative de Kubernetes. La plupart des administrateurs système sont habitués aux architectures impératives, où l'on effectue des actions qui modifient indirectement l'état du système, telles que le démarrage et l'arrêt de conteneurs sur un serveur donné. Dans une architecture déclarative, vous modifiez l'état du système et le système se met à jour pour se conformer au nouvel état. Par exemple, avec Kubernetes, vous définissez une ressource de pod qui spécifie qu'un certain conteneur doit s'exécuter dans des conditions spécifiques. Kubernetes trouve ensuite un serveur (un nœud) qui peut exécuter ce conteneur sous ces conditions spécifiques.

Les architectures déclaratives permettent l'auto-optimisation et l'autoréparation des systèmes qui sont plus faciles à gérer que les architectures impératives.

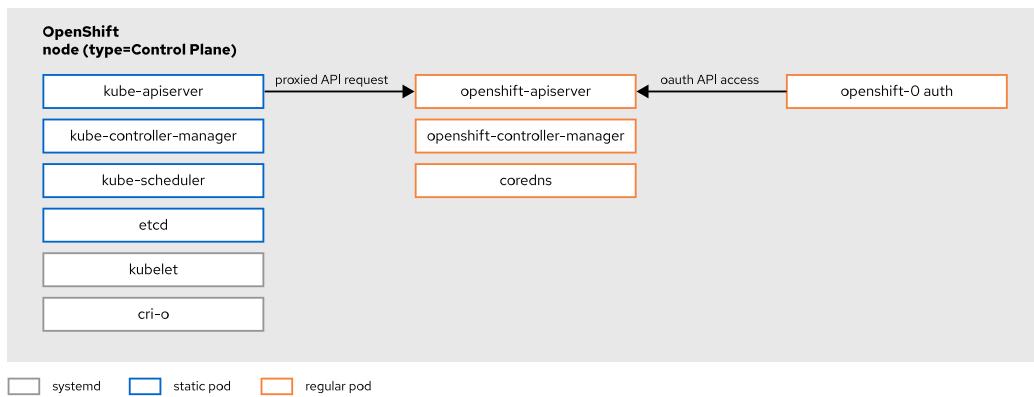
Kubernetes définit l'état de son cluster, y compris l'ensemble des applications déployées, comme un ensemble de ressources stockées dans la base de données etcd. Kubernetes exécute également des contrôleurs qui surveillent ces ressources et les compare à l'état actuel du cluster. Ces contrôleurs prennent toutes les mesures nécessaires pour concilier l'état du cluster avec l'état des ressources, en recherchant, par exemple, un nœud ayant une capacité de processeur suffisante pour démarrer un nouveau conteneur à partir d'une nouvelle ressource de pod.

Kubernetes fournit une API REST pour gérer ces ressources. Toutes les actions qu'un utilisateur OpenShift effectue à l'aide de l'interface de ligne de commande ou de la console Web sont exécutées en invoquant cette API REST.

## Présentation du plan de contrôle OpenShift

Un cluster Kubernetes se compose d'un ensemble de nœuds qui exécutent le service système kubelet et d'un moteur de conteneur. OpenShift exécute exclusivement le moteur de conteneur CRI-O. Certains nœuds sont des nœuds de plan de contrôle qui exécutent l'API REST, la base de données etcd et les contrôleurs de plateforme. OpenShift configure ses nœuds de plan de contrôle de sorte qu'ils ne puissent pas être programmés pour exécuter les pods d'application de l'utilisateur final et qu'ils soient dédiés à l'exécution des services du plan de contrôle. OpenShift planifie l'exécution des pods d'application de l'utilisateur final sur les nœuds de calcul.

Le graphique suivant fournit une vue d'ensemble d'un nœud de plan de contrôle OpenShift illustrant les principaux processus qui s'exécutent dans un nœud standard et dans un nœud de plan de contrôle, en tant que services système ou conteneurs.



**Figure 8.3: Architecture d'un nœud de plan de contrôle OpenShift**

En fonction des paramètres du nœud, l'agent `kubelet` démarre différents ensembles de pods statiques. Les pods statiques sont des pods qui n'ont pas besoin de se connecter au serveur de l'API pour démarrer. L'agent `kubelet` gère le cycle de vie du pod. Les pods statiques peuvent fournir des services de plan de contrôle, tels que le planificateur, ou des services de nœud, tels que le réseau défini par logiciel (SDN). OpenShift fournit des opérateurs qui créent des ressources de pod pour ces pods statiques, de sorte qu'ils soient surveillés comme des pods standard.

## Description des extensions OpenShift

De nombreuses fonctionnalités de Kubernetes dépendent de composants externes, tels que les contrôleurs d'entrée, les plug-ins de stockage, les plug-ins réseau et les plug-ins d'authentification. À l'instar des distributions Linux, il existe de nombreuses façons de créer une distribution Kubernetes en choisissant différents composants.

De nombreuses fonctionnalités de Kubernetes dépendent également des API d'extension, telles que le contrôle d'accès et l'isolement de réseau.

OpenShift est une distribution de Kubernetes qui fournit un grand nombre de ces composants déjà intégrés et configurés, et gérés par des opérateurs. OpenShift fournit également des applications préinstallées, telles qu'un registre d'images de conteneur et une console Web, gérées par des opérateurs.

OpenShift ajoute également à Kubernetes une série d'API d'extension et des ressources personnalisées. Par exemple, créez des configurations pour le processus Source-to-Image et des ressources d'itinéraire pour gérer l'accès externe au cluster.

Red Hat développe toutes les extensions en tant que projets Open Source et travaille avec la communauté Kubernetes élargie non seulement pour en faire des composants officiels de Kubernetes, mais également pour faire évoluer la plateforme Kubernetes afin de permettre une maintenance et une personnalisation plus faciles.

Avec OpenShift 3, ces extensions constituaient parfois des correctifs (ou des fourches) de Kubernetes en amont. Avec OpenShift 4 et les opérateurs, il s'agit d'extensions Kubernetes standard qui peuvent être ajoutées à toute distribution de Kubernetes.

## Présentation de la classe de stockage par défaut d'OpenShift

Contrairement à de nombreuses plateformes de conteneurs qui se concentrent sur des applications sans état natives dans le cloud, OpenShift prend également en charge des applications avec état qui ne suivent pas la méthodologie standard de *l'application à douze facteurs*. OpenShift prend en charge les applications avec état en offrant un ensemble exhaustif de capacités de stockage et d'opérateurs de prise en charge. OpenShift est fourni avec des plug-ins de stockage intégrés et des classes de stockage qui s'appuient sur la plateforme de virtualisation ou de cloud sous-jacente pour fournir un stockage déployé de manière dynamique.

Par exemple, si vous installez OpenShift sur Amazon Web Services (AWS), votre cluster OpenShift est préconfiguré avec une classe de stockage par défaut qui utilise automatiquement le service Amazon Elastic Block Store (EBS) pour déployer les volumes de stockage à la demande. Les utilisateurs peuvent déployer une application qui nécessite un stockage persistant, tel qu'une base de données, et OpenShift crée automatiquement un volume EBS pour héberger les données d'application.

Les administrateurs de cluster OpenShift peuvent définir par la suite des classes de stockage supplémentaires qui utilisent des niveaux de service EBS différents. Par exemple, vous pouvez avoir une classe de stockage pour le stockage haute performance qui supporte une vitesse élevée d'opérations d'entrées-sorties par seconde (IOPS) et une autre classe de stockage pour un stockage à faible coût et à faible performance. Les administrateurs de cluster peuvent alors autoriser uniquement certaines applications à utiliser la classe de stockage haute performance et configurer des applications d'archivage de données de manière à utiliser la classe de stockage à faible performance.



### Références

Vous trouverez des informations complémentaires dans la documentation produit de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse  
[https://access.redhat.com/documentation/en-us/  
openshift\\_container\\_platform/4.10/](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/)

### Application à douze facteurs

<https://12factor.net/>

## ► Quiz

# Description de l'architecture d'OpenShift

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les technologies d'orchestration de conteneur suivantes, sur laquelle est basé OpenShift ?**
  - a. Docker Swarm
  - b. Rancher
  - c. Kubernetes
  - d. Mesosphere Marathon
  - e. CoreOS Fleet
  
- ▶ 2. **Parmi les propositions suivantes, lesquelles s'appliquent à OpenShift Container Platform ? (Choisissez deux réponses.)**
  - a. OpenShift fournit un serveur OAuth qui authentifie les appels vers son API REST.
  - b. OpenShift nécessite un moteur de conteneur CRI-O.
  - c. Kubernetes suit une architecture déclarative, alors qu'OpenShift suit une architecture impérative plus traditionnelle.
  - d. Les API d'extension OpenShift s'exécutent en tant que services système.
  
- ▶ 3. **Parmi les serveurs suivants, lequel exécute des composants API Kubernetes ?**
  - a. Nœuds de calcul
  - b. Nœuds
  - c. Nœuds de plan de contrôle
  
- ▶ 4. **Parmi les composants suivants, lequel OpenShift ajoute-t-il à Kubernetes en amont ?**
  - a. La base de données etcd
  - b. Un moteur de conteneur
  - c. Un serveur de registre
  - d. Un planificateur
  - e. Le Kubelet

► 5. Parmi les phrases suivantes, laquelle s'applique à la prise en charge du stockage avec OpenShift ?

- a. Les utilisateurs ne peuvent stocker que des données persistantes dans la base de données etcd.
- b. Les utilisateurs ne peuvent déployer sur OpenShift que des applications natives dans le cloud conformes à la méthodologie de l'application à douze facteurs.
- c. Les administrateurs doivent configurer les plug-ins de stockage appropriés pour leurs fournisseurs de cloud.
- d. Les administrateurs doivent définir des volumes persistants avant qu'un utilisateur puisse déployer des applications qui nécessitent un stockage persistant.
- e. Les utilisateurs peuvent déployer des applications qui nécessitent un stockage persistant en utilisant la classe de stockage par défaut.

## ► Solution

# Description de l'architecture d'OpenShift

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les technologies d'orchestration de conteneur suivantes, sur laquelle est basé OpenShift ?**
  - a. Docker Swarm
  - b. Rancher
  - c. Kubernetes
  - d. Mesosphere Marathon
  - e. CoreOS Fleet
  
- ▶ 2. **Parmi les propositions suivantes, lesquelles s'appliquent à OpenShift Container Platform ? (Choisissez deux réponses.)**
  - a. OpenShift fournit un serveur OAuth qui authentifie les appels vers son API REST.
  - b. OpenShift nécessite un moteur de conteneur CRI-O.
  - c. Kubernetes suit une architecture déclarative, alors qu'OpenShift suit une architecture impérative plus traditionnelle.
  - d. Les API d'extension OpenShift s'exécutent en tant que services système.
  
- ▶ 3. **Parmi les serveurs suivants, lequel exécute des composants API Kubernetes ?**
  - a. Nœuds de calcul
  - b. Nœuds
  - c. Nœuds de plan de contrôle
  
- ▶ 4. **Parmi les composants suivants, lequel OpenShift ajoute-t-il à Kubernetes en amont ?**
  - a. La base de données etcd
  - b. Un moteur de conteneur
  - c. Un serveur de registre
  - d. Un planificateur
  - e. Le Kubelet

► 5. Parmi les phrases suivantes, laquelle s'applique à la prise en charge du stockage avec OpenShift ?

- a. Les utilisateurs ne peuvent stocker que des données persistantes dans la base de données etcd.
- b. Les utilisateurs ne peuvent déployer sur OpenShift que des applications natives dans le cloud conformes à la méthodologie de l'application à douze facteurs.
- c. Les administrateurs doivent configurer les plug-ins de stockage appropriés pour leurs fournisseurs de cloud.
- d. Les administrateurs doivent définir des volumes persistants avant qu'un utilisateur puisse déployer des applications qui nécessitent un stockage persistant.
- e. Les utilisateurs peuvent déployer des applications qui nécessitent un stockage persistant en utilisant la classe de stockage par défaut.

# Description des opérateurs de cluster

## Résultats

À la fin de cette section, vous devez pouvoir décrire ce qu'est un opérateur de cluster et son fonctionnement, ainsi que de nommer les principaux opérateurs de cluster.

## Présentation des opérateurs Kubernetes

Les opérateurs Kubernetes sont des applications qui invoquent l'API Kubernetes pour gérer les ressources Kubernetes. Comme pour toute application Kubernetes, vous déployez un opérateur en définissant des ressources Kubernetes, telles que des services et des déploiements faisant référence à l'image de conteneur de l'opérateur. Étant donné que les opérateurs, contrairement aux applications courantes, nécessitent un accès direct aux ressources Kubernetes, ils requièrent généralement des paramètres de sécurité personnalisés.

Les opérateurs définissent généralement des ressources personnalisées (CR) qui stockent leurs paramètres et leurs configurations. Un administrateur OpenShift gère un opérateur en modifiant ses ressources personnalisées. La syntaxe d'une ressource personnalisée est définie par une définition de ressources personnalisées (CRD).

La plupart des opérateurs gèrent une autre application ; par exemple, un opérateur qui gère un serveur de base de données. Dans ce cas, l'opérateur crée les ressources qui décrivent cette autre application à l'aide des informations de sa ressource personnalisée.

Le but d'un opérateur est généralement d'automatiser les tâches qu'un administrateur humain (ou un opérateur humain) doit effectuer pour déployer, mettre à jour et gérer une application.

## Présentation d'Operator Framework

Vous pouvez développer des opérateurs à l'aide du langage de programmation de votre choix. Techniquement, vous n'avez pas besoin d'un kit de développement logiciel (SDK) spécial pour développer un opérateur. Vous devez simplement pouvoir appeler des API REST et utiliser des secrets contenant des informations d'identification d'accès aux API Kubernetes.

Operator Framework est un kit d'outils Open Source qui permet de créer, de tester et d'empaqueter les opérateurs. Operator Framework simplifie ces tâches par rapport au codage direct des API Kubernetes de bas niveau en fournissant les composants suivants :

### Kit de développement logiciel de l'opérateur (SDK opérateur)

Il fournit un ensemble de bibliothèques GoLang et d'exemples de code source qui mettent en œuvre des modèles communs dans les applications d'opérateurs. Il fournit également une image de conteneur et des exemples de playbook qui vous permettent de développer des opérateurs à l'aide d'Ansible.

### Gestionnaire de cycle de vie d'opérateur (OLM)

Il fournit une application qui gère le déploiement, l'utilisation des ressources, les mises à jour et la suppression des opérateurs qui ont été déployés par le biais d'un catalogue d'opérateurs. Il s'agit d'un opérateur préinstallé avec OpenShift.

Operator Framework définit également un ensemble de pratiques recommandées pour l'implémentation des opérateurs et des CRD, ainsi qu'une méthode standard d'empaquetage d'un

manifeste d'opérateur, en tant qu'image de conteneur, qui permet à un opérateur d'être distribué à l'aide d'un catalogue d'opérateurs. Le catalogue d'opérateurs se présente le plus souvent sous la forme d'un serveur de registre d'images.

Une image de conteneur d'opérateurs qui suit les normes d'Operator Framework contient toutes les définitions de ressources nécessaires au déploiement de l'application de l'opérateur. De cette façon, l'OLM peut installer un opérateur automatiquement. Si un opérateur n'est pas créé et empaqueté en suivant les normes d'Operator Framework, l'OLM ne sera pas en mesure d'installer ni de gérer cet opérateur.

## Présentation d'OperatorHub

OperatorHub fournit une interface Web qui permet de détecter et de publier des opérateurs qui suivent les normes d'Operator Framework. Les opérateurs Open Source et les opérateurs commerciaux peuvent être publiés sur le hub de l'opérateur. Les images de conteneur d'opérateurs peuvent être hébergées dans différents registres d'images, comme quay.io.

## Présentation de Red Hat Marketplace

Red Hat Marketplace est une plateforme qui permet d'accéder à un ensemble géré d'opérateurs d'entreprise pouvant être déployés dans un cluster OpenShift ou Kubernetes. Les opérateurs disponibles dans Red Hat Marketplace ont suivi un processus de certification pour s'assurer que le logiciel suit les meilleures pratiques et que les conteneurs sont analysés à la recherche de vulnérabilités.

L'opérateur Red Hat Marketplace permet une intégration transparente entre un cluster OpenShift et Red Hat Marketplace. Cette intégration gère les mises à jour, et consolide la facturation et la création de rapports, ce qui simplifie le déploiement des opérateurs certifiés. Les fournisseurs proposent plusieurs options de tarification pour leurs opérateurs, telles que des versions d'essai gratuites, des éditions différentes et des remises pour les gros clients.

## Présentation des opérateurs de cluster OpenShift

Les opérateurs de cluster sont des opérateurs standard, sauf qu'ils ne sont pas gérés par l'OLM. Ils sont gérés par l'opérateur de version de cluster OpenShift, parfois appelé opérateur de premier niveau. Tous les opérateurs de cluster sont également appelés opérateurs de second niveau.

Les opérateurs de cluster OpenShift fournissent des API d'extension OpenShift et des services d'infrastructure tels que :

- le serveur OAuth, qui authentifie l'accès aux API de plan de contrôle et d'extension ;
- le serveur DNS principal, qui gère la découverte de services à l'intérieur du cluster ;
- la console Web, qui permet la gestion graphique du cluster ;
- le registre d'images interne, qui permet aux développeurs d'héberger des images de conteneur à l'intérieur du cluster, à l'aide de S2I ou d'un autre mécanisme ;
- la pile de surveillance, qui génère des indicateurs et des alertes concernant l'intégrité du cluster.

Certains opérateurs de cluster gèrent les paramètres des nœuds ou des plans de contrôle. Par exemple, avec Kubernetes en amont, vous éditez un fichier de configuration de nœud pour ajouter des plug-ins de stockage et réseau qui peuvent nécessiter des fichiers de configuration supplémentaires. OpenShift prend en charge les opérateurs qui gèrent les fichiers de configuration dans tous les nœuds et recharge les services de nœud qui sont affectés par les modifications apportées à ces fichiers.



### Important

OpenShift 4 déconseille d'utiliser les sessions SSH pour gérer les services de configuration de nœuds et système. Cela garantit que vous ne personnalisez pas les nœuds et qu'ils peuvent être ajoutés à un cluster ou en être supprimés sans risque. Vous êtes censé effectuer indirectement toutes les actions administratives en modifiant les ressources personnalisées, puis attendre que leurs opérateurs respectifs appliquent vos modifications.

## Exploration des opérateurs de cluster OpenShift

En général, un opérateur et son application gérée partagent le même projet. Dans le cas des opérateurs de cluster, ils se trouvent dans les projets `openshift-*`. Chaque opérateur de cluster définit une ressource personnalisée de type `ClusterOperator`. Les opérateurs de cluster gèrent le cluster lui-même, y compris le serveur d'API, la console Web ou la pile réseau. Chaque opérateur de cluster définit un ensemble de ressources personnalisées pour contrôler davantage ses composants. La ressource d'API `ClusterOperator` expose des informations telles que l'intégrité de la mise à jour ou la version du composant.

Les opérateurs sont visibles à partir de leur nom, par exemple, l'opérateur de cluster `console` fournit la console Web, l'opérateur de cluster `ingress` active les entrées et les routes. La liste suivante répertorie certains des opérateurs de cluster :

- `authentication`
- `cloud-credential`
- `cluster-autoscaler`
- `console`
- `dns`
- `image-registry`
- `ingress`
- `monitoring`
- `network`
- `openshift-apiserver`
- `openshift-controller-manager`
- `storage`



### Références

Vous trouverez des informations complémentaires dans la documentation produit de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse  
[https://access.redhat.com/documentation/en-us/  
openshift\\_container\\_platform/4.10/](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/)

#### Présentation d'Operator Framework

<https://blog.openshift.com/introducing-the-operator-framework/>

#### Premiers pas avec Red Hat Marketplace

<https://marketplace.redhat.com/en-us/documentation/getting-started>

## ► Quiz

# Description des opérateurs de cluster

Reliez les éléments ci-dessous aux éléments correspondants dans le tableau.

Catalogue d'opérateurs

Définition de ressources personnalisées

Gestionnaire de cycle de vie d'opérateur (OLM)

Image d'opérateur

OperatorHub

Opérateur

Red Hat Marketplace

SDK opérateur

Terminologie de l'opérateur	Name
Kit d'outils Open Source qui permet de créer, de tester et d'empaqueter les opérateurs.	
Référentiel qui permet de détecter et d'installer des opérateurs.	
Extension de l'API Kubernetes qui définit la syntaxe d'une ressource personnalisée.	
Artefact défini par Operator Framework que vous pouvez publier en vue de son utilisation par une instance OLM.	
Application qui gère les ressources Kubernetes.	
Application qui gère les opérateurs Kubernetes.	
Service Web public où vous pouvez publier des opérateurs compatibles avec OLM.	
Plateforme qui permet d'accéder à des logiciels certifiés empaquetés sous la forme d'opérateurs Kubernetes pouvant être déployés dans un cluster OpenShift.	

## ► Solution

# Description des opérateurs de cluster

Reliez les éléments ci-dessous aux éléments correspondants dans le tableau.

Terminologie de l'opérateur	Name
Kit d'outils Open Source qui permet de créer, de tester et d'empaqueter les opérateurs.	SDK opérateur
Référentiel qui permet de détecter et d'installer des opérateurs.	Catalogue d'opérateurs
Extension de l'API Kubernetes qui définit la syntaxe d'une ressource personnalisée.	Définition de ressources personnalisées
Artefact défini par Operator Framework que vous pouvez publier en vue de son utilisation par une instance OLM.	Image d'opérateur
Application qui gère les ressources Kubernetes.	Opérateur
Application qui gère les opérateurs Kubernetes.	Gestionnaire de cycle de vie d'opérateur (OLM)
Service Web public où vous pouvez publier des opérateurs compatibles avec OLM.	OperatorHub
Plateforme qui permet d'accéder à des logiciels certifiés empaquetés sous la forme d'opérateurs Kubernetes pouvant être déployés dans un cluster OpenShift.	Red Hat Marketplace

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Red Hat OpenShift Container Platform exécute les pods Kubernetes dans le moteur de conteneur CRI-O.
- RHOC 4 fournit des services en plus de Kubernetes, tels qu'un registre d'images de conteneur interne, du stockage et des fournisseurs de réseau.
- Les nœuds Red Hat OpenShift Container Platform reposent sur Red Hat Enterprise Linux CoreOS.
- RHOC 4 met en œuvre une journalisation et une surveillance centralisées.
- Les opérateurs créent des paquetages d'applications qui gèrent les ressources Kubernetes.
- Operator Lifecycle Manager (OLM) gère l'installation et la gestion des opérateurs.
- Operator Hub est un catalogue en ligne permettant de détecter des opérateurs.



# Vérification de la santé d'un cluster

### Objectif

Décrire les méthodes d'installation d'OpenShift et vérifier la santé d'un cluster nouvellement installé.

### Résultats

- Décrire le processus d'installation d'OpenShift, l'automatisation de la pile complète et les méthodes d'installation de l'infrastructure préexistante.
- Exécuter des commandes qui facilitent la résolution des problèmes, vérifier que les nœuds OpenShift sont intègres et résoudre les problèmes courants liés aux déploiements OpenShift et Kubernetes.
- Identifier les composants et les ressources du stockage persistant, et déployer une application qui utilise une revendication de volume persistant.

### Sections

- Description des méthodes d'installation (et quiz)
- Résolution des problèmes relatifs aux applications et aux clusters OpenShift (et exercice guidé)
- Présentation du stockage dynamique OpenShift (et exercice guidé)

# Description des méthodes d'installation

---

## Résultats

À la fin de cette section, vous devez pouvoir décrire le processus d'installation d'OpenShift, l'automatisation de la pile complète et les méthodes d'installation de l'infrastructure préexistante.

## Présentation des méthodes d'installation d'OpenShift

Red Hat OpenShift Container Platform propose deux méthodes d'installation principales :

### Automatisation de la pile complète

Avec cette méthode, le programme d'installation d'OpenShift met à disposition toutes les ressources de calcul, de stockage et de réseau d'un fournisseur de cloud ou de virtualisation. Vous fournissez au programme d'installation un minimum de données, telles que des informations d'identification à un fournisseur de cloud et la taille du cluster initial, puis le programme d'installation déploie un cluster OpenShift entièrement fonctionnel.

### Infrastructure préexistante

Avec cette méthode, vous configurez un ensemble de ressources de calcul, de stockage et de réseau et le programme d'installation d'OpenShift configure un cluster initial à l'aide de ces ressources. Vous pouvez utiliser cette méthode pour configurer un cluster OpenShift à l'aide de serveurs nus et de fournisseurs de services cloud ou de virtualisation qui ne sont pas pris en charge par la méthode d'automatisation de la pile complète.

Lorsque vous utilisez une infrastructure préexistante, vous devez fournir toute l'infrastructure et les ressources du cluster, y compris le noeud d'amorçage. Vous devez exécuter le programme d'installation pour générer les fichiers de configuration requis, puis exécuter à nouveau ce programme pour déployer un cluster OpenShift sur votre infrastructure.

À la sortie de Red Hat OpenShift Container Platform 4.10, l'ensemble des fournisseurs de cloud pris en charge pour la méthode d'automatisation de la pile complète comprenait Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure et Red Hat OpenStack Platform utilisant l'architecture Intel standard (x86). Les fournisseurs et architectures de virtualisation pris en charge pour l'automatisation de la pile complète sont notamment VMware, Red Hat Virtualization, IBM Power et IBM System Z.

Chaque version mineure du flux 4.x ajoute davantage de fonctionnalités et une prise en charge plus étendue des personnalisations, telles que la réutilisation de ressources cloud créées précédemment.

## Comparaison des méthodes d'installation d'OpenShift

Certaines fonctions d'OpenShift requièrent l'utilisation de la méthode d'automatisation de la pile complète, comme la mise à l'échelle automatique des clusters. Cependant, il est prévu que les prochaines versions soient moins exigeantes à cet égard.

Grâce à la méthode d'automatisation de la pile complète, tous les noeuds du nouveau cluster exécutent Red Hat Enterprise Linux CoreOS (RHEL CoreOS). Grâce à la méthode d'infrastructure préexistante, les noeuds de calcul peuvent être configurés à l'aide de Red Hat Enterprise Linux (RHEL), mais le plan de contrôle nécessite toujours RHEL CoreOS.

## Description du processus de déploiement

L'installation s'effectue en plusieurs étapes, dont la première consiste à créer une machine d'amorçage qui exécute Red Hat Enterprise Linux CoreOS à l'aide des ressources générées par le programme d'installation.

Voici le processus d'amorçage du cluster :

1. La machine d'amorçage démarre, puis commence à héberger les ressources distantes requises pour démarrer les machines du plan de contrôle.
2. Les machines du plan de contrôle récupèrent les ressources distantes de la machine d'amorçage et terminent le démarrage.
3. Les machines du plan de contrôle forment un cluster etcd.
4. La machine d'amorçage démarre un plan de contrôle Kubernetes temporaire à l'aide du cluster etcd que vous venez de créer.
5. Le plan de contrôle temporaire programme le plan de contrôle sur les machines du plan de contrôle.
6. Le plan de contrôle temporaire s'arrête pour céder la place au plan de contrôle.
7. Le nœud d'amorçage injecte des composants spécifiques à OpenShift dans le plan de contrôle.
8. Enfin, le programme d'installation détruit la machine d'amorçage.

Ce processus d'amorçage aboutit à un plan de contrôle OpenShift entièrement opérationnel, qui inclut le serveur d'API, les contrôleurs (tels que SDN) et le cluster etcd. Le cluster télécharge et configure ensuite les composants restants nécessaires au fonctionnement quotidien via l'opérateur de version de cluster, y compris la création automatisée de machines de calcul sur les plateformes prises en charge.

## Personnalisation d'une installation OpenShift

Le programme d'installation d'OpenShift permet de personnaliser légèrement le cluster initial déployé. La plupart des personnalisations s'effectuent après l'installation, notamment :

- la définition de classes de stockage personnalisées pour l'approvisionnement de stockage dynamique ;
- la modification des ressources personnalisées des opérateurs de cluster ;
- l'ajout de nouveaux opérateurs à un cluster ;
- la définition de nouveaux ensembles de machines.



## Références

Pour plus d'informations sur les différentes méthodes d'installation, reportez-vous à la documentation *Installing* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/installing/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/installing/index)

Pour plus d'informations sur l'infrastructure déployée par le programme d'installation, visionnez la vidéo *OpenShift 4.x Installation - Quick Overview (IPI Installation)* de Red Hat OpenShift Container Platform 4.10 à l'adresse

<https://youtu.be/uBsib4cuAl>

Pour plus d'informations sur l'infrastructure déployée par l'utilisateur, visionnez la vidéo *OpenShift 4 User Provisioned Infrastructure with VMware vSphere* de Red Hat OpenShift Container Platform 4.10 à l'adresse

<https://youtu.be/TsAJEEDv-gg>

## ► Quiz

# Description des méthodes d'installation

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les méthodes d'installation suivantes, laquelle nécessite l'utilisation du programme d'installation d'OpenShift pour configurer les nœuds de plan de contrôle et de calcul ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
  
- ▶ 2. **Parmi les méthodes d'installation suivantes, laquelle permet de configurer des nœuds à l'aide de Red Hat Enterprise Linux ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
  
- ▶ 3. **Parmi les méthodes d'installation suivantes, laquelle permet d'utiliser un fournisseur de virtualisation non pris en charge aux dépens de certaines fonctions d'OpenShift ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
  
- ▶ 4. **Quelle méthode d'installation permet d'utiliser plusieurs fournisseurs de cloud pris en charge avec un minimum d'efforts ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
  
- ▶ 5. **Parmi les méthodes d'installation suivantes, laquelle permet une personnalisation poussée des paramètres du cluster en fournissant des données au programme d'installation d'OpenShift ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre

## ► Solution

# Description des méthodes d'installation

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Parmi les méthodes d'installation suivantes, laquelle nécessite l'utilisation du programme d'installation d'OpenShift pour configurer les nœuds de plan de contrôle et de calcul ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
- ▶ 2. **Parmi les méthodes d'installation suivantes, laquelle permet de configurer des nœuds à l'aide de Red Hat Enterprise Linux ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
- ▶ 3. **Parmi les méthodes d'installation suivantes, laquelle permet d'utiliser un fournisseur de virtualisation non pris en charge aux dépens de certaines fonctions d'OpenShift ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
- ▶ 4. **Quelle méthode d'installation permet d'utiliser plusieurs fournisseurs de cloud pris en charge avec un minimum d'efforts ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre
- ▶ 5. **Parmi les méthodes d'installation suivantes, laquelle permet une personnalisation poussée des paramètres du cluster en fournissant des données au programme d'installation d'OpenShift ?**
  - a. Automatisation de la pile complète
  - b. Infrastructure préexistante
  - c. Les deux méthodes
  - d. Ni l'une, ni l'autre

# Résolution des problèmes relatifs aux applications et aux clusters OpenShift

---

## Résultats

À la fin de cette section, vous devez pouvoir exécuter des commandes qui facilitent la résolution des problèmes, vérifier que les nœuds OpenShift sont intègres et résoudre les problèmes courants liés aux déploiements OpenShift et Kubernetes.

## Résolution des problèmes courants liés à un cluster OpenShift

La plupart des opérations de résolution de problèmes du cluster OpenShift sont très similaires à la résolution des problèmes liés aux déploiements d'applications, car la plupart des composants de Red Hat OpenShift 4 sont des opérateurs, et les opérateurs sont des applications Kubernetes. Pour chaque opérateur, vous pouvez identifier le projet dans lequel il réside, le déploiement qui gère l'application de l'opérateur et ses pods. Si cet opérateur est doté de paramètres de configuration nécessitant d'être modifiés, vous pouvez identifier la ressource personnalisée (CR), ou parfois le mappage de configuration ou la ressource secrète qui stocke ces paramètres.

La plupart des opérateurs OpenShift gèrent des applications qui sont également déployées à partir de ressources d'API de charge de travail Kubernetes standard, telles que des ensembles de démons et des déploiements. Le rôle de l'opérateur consiste généralement à créer ces ressources et à les maintenir synchronisées avec la CR.

Cette section se concentre d'abord sur les problèmes de cluster n'étant pas directement liés aux opérateurs ou aux déploiements d'applications ; plus loin dans cette section, vous apprendrez à résoudre les problèmes liés aux déploiements d'applications.

## Vérification de l'intégrité des nœuds OpenShift

Les commandes suivantes affichent des informations sur l'état et l'intégrité des nœuds d'un cluster OpenShift :

`oc get nodes`

Affiche une colonne avec l'état de chaque nœud. Si un nœud n'est pas Ready, il ne peut alors pas communiquer avec le plan de contrôle OpenShift et il est effectivement obsolète pour le cluster.

`oc adm top nodes`

Affiche l'utilisation actuelle du processeur et de la mémoire de chaque nœud. Il s'agit des numéros d'utilisation réels, et non des demandes de ressources que le planificateur OpenShift considère comme la capacité disponible et utilisée du nœud.

`oc describe node my-node-name`

Affiche les ressources disponibles et utilisées du point de vue du planificateur, ainsi que d'autres informations. Recherchez les titres « Capacity », « Allocatable » et « Allocated Resources » dans la sortie. Le titre « Conditions » indique si la mémoire ou le disque du nœud sont sollicités ou si toute autre condition empêche le nœud de démarrer de nouveaux conteneurs.

## Examen de la ressource de la version du cluster

Le programme d'installation d'OpenShift crée un répertoire auth contenant les fichiers `kubeconfig` et `kubeadm-password`. Exécutez la commande `oc login` pour vous connecter au cluster avec l'utilisateur `kubeadm`. Le mot de passe de l'utilisateur `kubeadm` se trouve dans le fichier `kubeadm-password`.

```
[user@host ~]$ oc login -u kubeadm -p MMTUc-TnXjo-NFyh3-aeWmc \
>   https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

`ClusterVersion` est une ressource personnalisée qui contient des informations détaillées sur le cluster, telles que les canaux de mise à jour, l'état des opérateurs du cluster, ainsi que la version du cluster (par exemple 4.10.3). Utilisez cette ressource pour déclarer la version du cluster que vous souhaitez exécuter. La définition d'une nouvelle version pour le cluster indique à l'opérateur `cluster-version` de mettre à niveau le cluster vers cette version.

Vous pouvez récupérer la version du cluster pour vérifier qu'elle exécute la version souhaitée, mais également pour vous assurer que le cluster utilise le bon canal d'abonnement.

- Exécutez `oc get clusterversion` pour récupérer la version du cluster. La sortie répertorie la version, y compris les versions mineures, la disponibilité du cluster pour une version donnée et l'état global du cluster.

```
[user@host ~]$ oc get clusterversion
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE    STATUS
version   4.10.3   True       False        35d      Cluster version is 4.10.3
```

- Exécutez `oc describe clusterversion` pour obtenir plus d'informations détaillées sur l'état du cluster.

```
[user@host ~]$ oc describe clusterversion
Name:           version
Namespace:
Labels:         <none>
Annotations:   <none>
API Version:  config.openshift.io/v1
Kind:          ClusterVersion
...output omitted...
Spec:
  Channel:     stable-4.10  ①
  Cluster ID: f33267f8-260b-40c1-9cf3-ecc406ce035e ②
Status:
  Available Updates: <nil> ③
  Channels:
    candidate-4.10
    candidate-4.11
    eus-4.10
    fast-4.10
    stable-4.10
  Image:      quay.io/openshift-release-dev/ocp-release...
  URL:       https://access.redhat.com/errata/RHBA-2022:0811
  Version:   4.10.4
```

```

Conditions:
Last Transition Time: 2022-03-15T12:39:02Z
Message: Done applying 4.10.3 ④
Status: True
Type: Available
...output omitted...
Desired:
Channels:
candidate-4.10
candidate-4.11
eus-4.10
fast-4.10
stable-4.10
...output omitted...
History:
Completion Time: 2022-03-15T12:39:02Z ⑤
...output omitted...
State: Completed ⑥
Verified: false
Version: 4.10.3
Observed Generation: 2
...output omitted...

```

- ❶ Affiche la version du cluster et son canal. En fonction de votre abonnement, le canal peut être différent.
- ❷ Affiche l'identifiant unique du cluster. Red Hat utilise cet identifiant pour déterminer les clusters et les droits du cluster.
- ❸ Cette entrée liste les images disponibles pour mettre à jour le cluster.
- ❹ Cette entrée liste l'historique. La sortie indique qu'une mise à jour est terminée.
- ❺ Cette entrée s'affiche lorsque le cluster a déployé la version indiquée par l'entrée Version.
- ❻ Cette entrée indique que la version a bien été déployée. Utilisez cette entrée pour déterminer si le cluster est intégrer.

## Examen des opérateurs de cluster

Les *opérateurs de cluster* d'OpenShift Container Platform sont des opérateurs de niveau supérieur qui gèrent le cluster. Ils sont responsables des composants principaux, tels que le serveur d'API, la console Web, le stockage ou le SDN. Leurs informations sont accessibles par le biais de la ressource `ClusterOperator`, qui vous permet d'accéder à l'aperçu de tous les opérateurs de cluster ou aux informations détaillées sur un opérateur donné.

Exécutez `oc get clusteroperators` pour récupérer la liste de tous les opérateurs du cluster :

[user@host ~]\$ oc get clusteroperators						
NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE	
authentication	4.10.3	True	False	False	35d ①	
baremetal	4.10.3	True	False	False	35d	
cloud-controller-manager	4.10.3	True	False	False	35d	
cloud-credential	4.10.3	True	False	False	35d	
cluster-autoscaler	4.10.3	True	False	False	35d	

**chapitre 9 |** Vérification de la santé d'un cluster

config-operator	4.10.3	True	False	False	35d
console	4.10.3	True	False	False	3h22m
csi-snapshot-controller	4.10.3	True	False	False	35d
dns	4.10.3	True	False	False	35d
etcd	4.10.3	True	False	False	35d
image-registry	4.10.3	True	False	False	35d
<i>...output omitted...</i>					

- ➊ Chaque ligne décrit un opérateur de cluster.

Le champ NAME indique le nom de l'opérateur. Cet opérateur est responsable de la gestion de l'authentification.

Le champ AVAILABLE indique que l'opérateur d'authentication a bien été déployé et qu'il peut être utilisé dans le cluster. Notez qu'un opérateur de cluster peut renvoyer l'état available (disponible), même s'il est dégradé. Un opérateur est dégradé lorsque son état actuel ne correspond pas à son état souhaité sur une période donnée. Par exemple, si l'opérateur requiert trois pods en cours d'exécution, mais qu'un pod se bloque, l'opérateur est disponible, mais dans un état dégradé.

Le champ PROGRESSING indique si un opérateur est mis à jour vers une version plus récente de l'opérateur de niveau supérieur. Si de nouvelles ressources sont déployées par l'opérateur cluster version, les colonnes sont définies sur True.

Le champ DEGRADED renvoie l'intégrité de l'opérateur. L'entrée est définie sur True si l'opérateur rencontre une erreur qui l'empêche de fonctionner correctement. Il se peut que les services de l'opérateur soient toujours disponibles, mais que toutes les exigences ne soient pas satisfaites. Cela peut indiquer que l'opérateur va échouer et nécessiter l'intervention de l'utilisateur.

## Affichage des journaux des nœuds OpenShift

La plupart des composants d'infrastructure d'OpenShift sont des conteneurs à l'intérieur de pods ; vous pouvez afficher leurs journaux de la même façon que vous affichez les journaux pour n'importe quelle application d'utilisateur final. Certains de ces conteneurs sont créés par le Kubelet, et donc invisibles pour la plupart des distributions de Kubernetes, mais les opérateurs de cluster OpenShift créent des ressources de pod pour ceux-ci.

Un nœud OpenShift basé sur Red Hat Enterprise Linux CoreOS exécute très peu de services locaux qui nécessiteraient un accès direct à un nœud pour inspecter leur état. La plupart des services système de Red Hat Enterprise Linux CoreOS s'exécutent en tant que conteneurs. Les principales exceptions sont le moteur de conteneur CRI-O et le Kubelet, qui sont des unités Systemd. Pour afficher ces journaux, utilisez la commande oc adm node-logs, comme illustré dans les exemples suivants :

```
[user@host ~]$ oc adm node-logs -u crio my-node-name
[user@host ~]$ oc adm node-logs -u kubelet my-node-name
```

Vous pouvez également afficher tous les journaux de journal d'un nœud :

```
[user@host ~]$ oc adm node-logs my-node-name
```

## Ouverture d'une invite de shell sur un nœud OpenShift

Les administrateurs qui gèrent Red Hat OpenShift Cluster Platform 3 et d'autres distributions de Kubernetes ouvrent fréquemment des sessions SSH sur leurs nœuds pour inspecter l'état du plan de contrôle et du moteur de conteneur, ou pour modifier les fichiers de configuration. Bien que cela puisse encore être fait, ce n'est plus recommandé avec Red Hat OpenShift Cluster Platform 4.

Si vous installez votre cluster à l'aide de la méthode d'automatisation de la pile complète, alors vos nœuds de cluster ne sont pas directement accessibles depuis Internet, car ils se trouvent sur un réseau privé virtuel, qu'AWS appelle le cloud privé virtuel (VPC). Pour ouvrir des sessions SSH, un serveur bastion sur le même VPC que votre cluster, auquel est également attribuée une adresse IP publique, est requis. La création d'un serveur bastion dépend de votre fournisseur de cloud et se trouve hors de la portée de ce cours.

La commande `oc debug node` permet d'ouvrir une invite de shell dans n'importe quel nœud de votre cluster. Cette invite provient d'un conteneur d'outils spécifique qui monte le système de fichiers root du nœud dans le dossier `/host` et vous permet d'inspecter tous les fichiers du nœud.

Pour exécuter des commandes locales directement à partir du nœud dans une session `oc debug node`, vous devez démarrer un shell chroot dans le dossier `/host`. Ensuite, vous pouvez inspecter les systèmes de fichiers locaux du nœud, l'état de ses services `systemd`, et effectuer d'autres tâches qui autrement nécessiteraient une session SSH. Voici un exemple de session `oc debug node` :

```
[user@host ~]$ oc debug node/my-node-name
...output omitted...
sh-4.4# chroot /host
sh-4.4# systemctl is-active kubelet
active
```

Pour fonctionner, une session shell démarrée à partir de la commande `oc debug node` dépend du plan de contrôle OpenShift. Elle utilise la même technologie de mise sous tunnel que celle qui permet d'ouvrir une invite de shell à l'intérieur d'un pod en cours d'exécution (voir la commande `oc rsh` plus loin dans cette section). La commande `oc debug node` n'est pas basée sur les protocoles SSH ou RSH.

Si votre plan de contrôle ne fonctionne pas, si votre nœud n'est pas prêt ou si, pour quelque raison que ce soit, votre nœud ne peut pas communiquer avec le plan de contrôle, alors vous ne pouvez pas recourir à la commande `oc debug node` et vous aurez besoin d'un hôte bastion.



### Mise en garde

Soyez attentif lorsque vous utilisez la commande `oc debug node`. Certaines actions peuvent rendre votre nœud inutilisable, telles que l'arrêt de Kubelet, et vous ne pouvez pas y remédier en utilisant uniquement les commandes `oc`.

## Résolution des problèmes du moteur de conteneur

Depuis une session `oc debug node`, utilisez la commande `crlctl` pour obtenir des informations de bas niveau sur tous les conteneurs locaux exécutés sur le nœud. Vous ne pouvez pas utiliser la commande `podman` pour cette tâche, car elle ne dispose pas de la visibilité sur les conteneurs créés par CRI-O. L'exemple suivant répertorie tous les conteneurs exécutés sur un nœud. La commande `oc describe node` fournit les mêmes informations, mais les organise par pod et non pas par conteneur.

```
[user@host ~]$ oc debug node/my-node-name
...output omitted...
sh-4.4# chroot /host
sh-4.4# crictl ps
...output omitted...
```

## Résolution de problèmes lors du déploiement de l'application

Vous pouvez généralement ignorer les différences entre les déploiements Kubernetes et les configurations de déploiement OpenShift lors de la résolution de problèmes liés aux applications. Les scénarios de défaillance courants et les moyens de les résoudre sont essentiellement les mêmes.

De nombreux scénarios seront décrits dans les chapitres ultérieurs de ce cours, tels que les pods ne pouvant pas être planifiés. Cette section se concentre sur les scénarios courants qui s'appliquent aux applications génériques, et les mêmes scénarios s'appliquent généralement aussi aux opérateurs.

### Résolution des problèmes de démarrage de pod

Il est courant qu'OpenShift crée un pod et que ce pod n'établit jamais l'état `Running`. Cela signifie qu'OpenShift n'a pas pu démarrer les conteneurs à l'intérieur de ce pod. Commencez par résoudre les problèmes à l'aide des commandes `oc get pod` et `oc status` pour vérifier si vos pods et conteneurs sont en cours d'exécution. À un moment donné, les pods sont en état d'erreur, par exemple `ErrImagePull` ou `ImagePullBackOff`.

Lorsque cela se produit, la première étape consiste à répertorier les événements du projet en cours à l'aide de la commande `oc get events`. Si votre projet contient de nombreux pods, vous pouvez alors obtenir une liste d'événements filtrés par pod à l'aide de la commande `oc describe pod`. Vous pouvez également exécuter des commandes similaires à `oc describe` pour filtrer les événements par déploiements et configurations de déploiement.

### Résolution des problèmes de pods en cours d'exécution et terminés

Il est également courant qu'OpenShift crée un pod et que, pour une courte durée, aucun problème n'est rencontré. Le pod passe à l'état `Running`, ce qui signifie qu'au moins un de ses conteneurs a commencé à s'exécuter. Par la suite, une application exécutée à l'intérieur d'un des conteneurs de pods cesse de fonctionner. Elle peut s'interrompre ou renvoyer des messages d'erreur aux demandes de l'utilisateur.

Si l'application est gérée par un déploiement correctement conçu, elle doit inclure des sondes d'intégrité qui finiront par interrompre l'application et arrêter son conteneur. Si cela se produit, OpenShift essaie de redémarrer le conteneur plusieurs fois. Si l'interruption de l'application continue, à cause des sondes d'intégrité ou pour d'autres raisons, le pod sera laissé à l'état `CrashLoopBackOff`.

Un conteneur en cours d'exécution, même pendant une très courte période, génère des journaux. Ces journaux ne sont pas supprimés lorsque le conteneur s'interrompt. La commande `oc logs` affiche les journaux de n'importe quel conteneur à l'intérieur d'un pod. Si le pod contient un seul conteneur, la commande `oc logs` requiert uniquement le nom du pod.

```
[user@host ~]$ oc logs my-pod-name
```

Si le pod contient plusieurs conteneurs, la commande `oc logs` requiert l'option `-c`.

```
[user@host ~]$ oc logs my-pod-name -c my-container-name
```

L'interprétation des journaux d'application nécessite une connaissance spécifique de cette application donnée. Si tout se passe bien, l'application fournit des messages d'erreur clairs qui peuvent vous aider à trouver le problème.

## Présentation de la journalisation agrégée d'OpenShift

Red Hat OpenShift Container Platform 4 fournit le sous-système de journalisation de cluster, basé sur Elasticsearch, Fluent ou Rsyslog, et Kibana, qui agrège les journaux à partir du cluster et de ses conteneurs.

Le déploiement et la configuration du sous-système de journalisation de cluster OpenShift via son opérateur dépassent le cadre de ce cours. Pour plus d'informations, reportez-vous à la section références à la fin de cette section.

## Création de pods de résolution des problèmes

Si vous n'êtes pas sûr que vos problèmes soient liés à l'image de conteneur d'application, ou aux paramètres qu'elle obtient de ses ressources OpenShift, la commande `oc debug` est très utile. Cette commande crée un pod à partir d'un pod existant, d'une configuration de déploiement, d'un déploiement ou de toute autre ressource de l'API de charges de travail.

Le nouveau pod exécute un shell interactif au lieu du point d'entrée par défaut de l'image de conteneur. Les sondes d'intégrité sont également désactivées lorsqu'il s'exécute. De cette façon, vous pouvez facilement vérifier les variables d'environnement, l'accès réseau aux autres services et les permissions à l'intérieur du pod.

Les options de ligne de commande de la commande `oc debug` vous permettent de spécifier les paramètres que vous ne souhaitez pas cloner. Par exemple, vous pouvez modifier l'image de conteneur ou spécifier un ID utilisateur fixe. Certains paramètres peuvent nécessiter des priviléges d'administrateur de cluster.

Un scénario courant consiste à créer un pod à partir d'un déploiement, mais exécuté en tant qu'utilisateur root, prouvant ainsi que le déploiement fait référence à une image de conteneur qui n'a pas été conçue pour être exécutée sous les politiques de sécurité par défaut d'OpenShift :

```
[user@host ~]$ oc debug deployment/my-deployment-name --as-root
```

## Modification d'un conteneur en cours d'exécution

Étant donné que les images de conteneur sont immuables et que les conteneurs sont supposés être éphémères, il n'est pas recommandé de modifier les conteneurs en cours d'exécution. Cependant, il peut arriver que ces modifications facilitent la résolution des problèmes liés aux applications. Une fois que vous avez essayé de modifier un conteneur en cours d'exécution, n'oubliez pas d'appliquer les mêmes modifications à l'image de conteneur et à ses ressources d'application, puis vérifiez que les corrections permanentes fonctionnent comme prévu.

## chapitre 9 | Vérification de la santé d'un cluster

Les commandes suivantes permettent d'apporter des modifications à des conteneurs en cours d'exécution. Elles supposent toutes que les pods contiennent un seul conteneur. Si tel n'est pas le cas, vous devez ajouter l'option `-c my-container-name`.

`oc rsh my-pod-name`

Ouvre un shell à l'intérieur d'un pod pour exécuter des commandes shell de manière interactive et non interactive.

`oc cp /local/path my-pod-name:/container/path`

Copie des fichiers locaux vers un emplacement à l'intérieur d'un pod. Vous pouvez également inverser des arguments et copier des fichiers depuis l'intérieur d'un pod vers votre système de fichiers local. Voir également la commande `oc rsync` pour copier plusieurs fichiers à la fois.

`oc port-forward my-pod-name local-port:remote-port`

Crée un tunnel TCP à partir de `local-port` sur votre station de travail vers le `local-port` du pod. Le tunnel est actif tant que vous continuez à exécuter `oc port-forward`. Cela vous permet d'obtenir un accès réseau au pod sans l'exposer via une route. Dans la mesure où le tunnel démarre sur votre localhost, d'autres machines ne peuvent pas y accéder.

## Résolution de problèmes de commandes CLI OpenShift

Parfois, vous ne comprenez pas pourquoi une commande `oc` échoue et vous devez résoudre les problèmes liés à ses actions de bas niveau pour en trouver la cause. Vous avez peut-être besoin de savoir ce qu'une invocation donnée de la commande `oc` réalise en arrière-plan, afin de pouvoir répliquer ce comportement avec un outil d'automatisation exécutant des demandes d'API OpenShift et Kubernetes, tel que des playbooks Ansible utilisant le module `k8s`.

L'option `--loglevel level` affiche les demandes d'API OpenShift à partir du niveau 6. Au fur et à mesure que vous augmentez le niveau, jusqu'au niveau 10, davantage d'informations sur ces demandes sont ajoutées, telles que leurs en-têtes de requête HTTP et les corps de réponse. Le niveau 10 inclut également une commande `curl` pour répliquer chaque requête.

Vous pouvez essayer ces deux commandes, depuis n'importe quel projet, et comparer leurs résultats.

```
[user@host ~]$ oc get pods --loglevel 6
```

```
[user@host ~]$ oc get pods --loglevel 10
```

Parfois, vous avez uniquement besoin du jeton d'authentification que la commande `oc` utilise pour authentifier les demandes de l'API OpenShift. Avec ce jeton, un outil d'automatisation peut faire des demandes d'API OpenShift comme s'il était connecté en tant qu'utilisateur. Pour récupérer votre jeton, utilisez l'option `-t` de la commande `oc whoami`:

```
[user@host ~]$ oc whoami -t
```



## Références

Pour plus d'informations sur les événements OpenShift, reportez-vous à la section *Viewing system event information in an OpenShift Container Platform cluster* du chapitre *Working with clusters* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#nodes-containers-events](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#nodes-containers-events)

Pour plus d'informations sur la manière de copier des fichiers sur des conteneurs en cours d'exécution, reportez-vous à la section *Copying files to or from an OpenShift Container Platform container* du chapitre *Working with containers* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#nodes-containers-copying-files](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#nodes-containers-copying-files)

Pour plus d'informations sur la manière d'exécuter des commandes sur des conteneurs en cours d'exécution, reportez-vous à la section *Executing remote commands in an OpenShift Container Platform container* du chapitre *Working with containers* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#nodes-containers-remote-commands](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#nodes-containers-remote-commands)

Pour plus d'informations sur la manière de transférer des ports locaux sur des conteneurs en cours d'exécution, reportez-vous à la section *Using port forwarding to access applications in a container* du chapitre *Working with containers* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#nodes-containers-port-forwarding](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#nodes-containers-port-forwarding)

Pour plus d'informations sur la journalisation agrégée, reportez-vous à la documentation *Logging* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/logging/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/logging/index)

## Ressource personnalisée ClusterOperator

<https://github.com/openshift/cluster-version-operator>

## ► Exercice guidé

# Résolution des problèmes relatifs aux applications et aux clusters OpenShift

Dans cet exercice, vous allez exécuter des commandes qui vous aideront à résoudre les problèmes courants liés au plan de contrôle OpenShift et aux déploiements d'applications.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Inspecter l'état général d'un cluster OpenShift.
- Inspecter les services et les pods locaux qui s'exécutent dans un nœud de calcul OpenShift.
- Diagnostiquer et résoudre les problèmes liés au déploiement d'une application.

## Avant De Commencer



### Important

La première fois que vous démarrez votre environnement de formation, les clusters OpenShift mettent un peu plus de temps à devenir entièrement disponibles. La commande `lab` au début de chaque exercice peut échouer si les clusters ne sont pas entièrement disponibles.

Si vous essayez d'accéder à votre cluster à l'aide de la commande `oc` ou de la console Web sans exécuter la commande `lab` au préalable, votre cluster ne sera peut-être pas encore disponible. Le cas échéant, attendez quelques minutes et réessayez.

Vous pouvez exécuter le script `wait.sh` sur la machine `utility`, qui attend que votre cluster OpenShift soit prêt à accepter les demandes d'authentification et d'API de la part des clients distants.

```
[student@workstation ~]$ ssh lab@utility
[lab@utility ~]$ ./wait.sh
```

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande permet de s'assurer que l'API du cluster est accessible et crée les fichiers de ressources que vous utiliserez au cours de l'activité. Elle crée également le projet `install-troubleshoot` avec une application que vous allez diagnostiquer et corriger au cours de cet exercice.

```
[student@workstation ~]$ lab install-troubleshoot start
```

## Instructions

- 1. Connectez-vous au cluster OpenShift et examinez l'état de vos nœuds de cluster.

- 1.1. Procurez-vous le fichier de configuration de la salle de classe, accessible à l'adresse /usr/local/etc/ocp4.config.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 1.2. Connectez-vous au cluster en tant qu'utilisateur kubeadmin. Lorsque vous y êtes invité, acceptez le certificat non sécurisé.

```
[student@workstation ~]$ oc login -u kubeadmin -p ${RHT_OCP4_KUBEADM_PASSWD} \
> https://api.ocp4.example.com:6443
The server uses a certificate signed by an unknown authority.
You can bypass the certificate check, but any data you send to the server could be
intercepted by others.
Use insecure connections? (y/n): y

Login successful.
...output omitted...
```

- 1.3. Vérifiez que tous les nœuds sur votre cluster sont prêts.

```
[student@workstation ~]$ oc get nodes
NAME      STATUS    ROLES          AGE     VERSION
master01   Ready     master,worker  36d    v1.23.3+e419edf
master02   Ready     master,worker  36d    v1.23.3+e419edf
master03   Ready     master,worker  36d    v1.23.3+e419edf
```

- 1.4. Vérifiez si l'un de vos nœuds s'apprête à utiliser la totalité de la capacité processeur et mémoire disponible.

Répétez la commande suivante plusieurs fois pour vous assurer que vous voyez l'utilisation réelle de processeur et de mémoire de vos nœuds. Les nombres que vous voyez doivent changer légèrement chaque fois que vous répétez la commande.

```
[student@workstation ~]$ oc adm top node
NAME      CPU(cores)   CPU%    MEMORY(bytes)  MEMORY%
master01   743m        21%    9002Mi        60%
master02   747m        21%    6112Mi        41%
master03   1047m       29%    10150Mi       68%
```

- 1.5. Utilisez la commande oc describe pour vérifier que toutes les conditions pouvant révéler un problème sont fausses.

```
[student@workstation ~]$ oc describe node master01
...output omitted...
Conditions:
  Type            Status  ...  Message
  ----           -----  ...  -----
  MemoryPressure  False   ...  kubelet has sufficient memory available
  DiskPressure    False   ...  kubelet has no disk pressure
```

**chapitre 9 |** Vérification de la santé d'un cluster

```
PIDPressure      False    ... kubelet has sufficient PID available
Ready           True     ... kubelet is posting ready status
Addresses:
...output omitted...
```

- 2. Examinez les journaux de l'opérateur de registre interne, du serveur de registre interne et du Kubelet d'un nœud.

- 2.1. Répertoriez tous les pods au sein du projet **openshift-image-registry**, puis identifiez le pod qui exécute l'opérateur et le pod qui exécute le serveur de registre interne.

```
[student@workstation ~]$ oc get pod -n openshift-image-registry
NAME                                     READY   STATUS    ...
cluster-image-registry-operator-6f7784dc86-qq258  1/1    Running   ...
image-pruner-27506880-dcn8n                 0/1    Completed  ...
image-pruner-27508320-2vfph                0/1    Completed  ...
image-registry-867979b75b-xk4ns              1/1    Running   ...
...output omitted...
```

- 2.2. Suivez les journaux du pod d'opérateur (**cluster-image-registry-operator-xxx**). Votre sortie peut être différente de celle de l'exemple suivant.

```
[student@workstation ~]$ oc logs --tail 3 -n openshift-image-registry \
>   cluster-image-registry-operator-564bd5dd8f-s46bz
I0421 01:30:11.357218      1 generator.go:60] object *v1.DaemonSet,
Namespace=openshift-image-registry, ...
I0421 01:30:18.290064      1 recorder_logging.go:44] &Event{ObjectMeta:
{dummy.16e7c534a15072a6 dummy ...
I0421 01:30:18.290674      1 generator.go:60] object *v1.DaemonSet,
Namespace=openshift-image-registry, ...
...output omitted...
```

- 2.3. Suivez les journaux du pod du serveur de registre d'image (**image-registry-xxx** de la sortie de la commande `oc get pod` exécutée précédemment). Votre sortie peut être différente de celle de l'exemple suivant.

```
[student@workstation ~]$ oc logs --tail 1 -n openshift-image-registry \
>   image-registry-794dfc7978-w7w69
I0421 01:30:11.357218      1 generator.go:60] object *v1.DaemonSet,
Namespace=openshift-image-registry, ...
I0421 01:30:18.290064      1 recorder_logging.go:44] &Event{ObjectMeta:
{dummy.16e7c534a15072a6 dummy ...
...output omitted...
```

- 2.4. Suivez les journaux du Kubelet du même nœud pour lequel vous avez inspecté l'utilisation du processeur et de la mémoire à l'étape précédente. Votre sortie peut être différente de celle de l'exemple suivant.

```
[student@workstation ~]$ oc adm node-logs --tail 1 -u kubelet master01
-- Logs begin at Tue 2022-03-15 12:21:41 UTC, end at Thu 2022-04-21 02:17:38 UTC.
--
Apr 20 17:30:28.172354 master01 systemd[1]: kubelet.service: Consumed 2h 12min
42.989s CPU time
-- Logs begin at Tue 2022-03-15 12:21:41 UTC, end at Thu 2022-04-21 02:17:38 UTC.
--
Apr 21 02:17:05.808392 master01 hyperkube[1844]: I0421 02:17:05.808374      1844
kubelet_getters.go:176] ...
```

- 3. Démarrez une session shell sur le même nœud que vous avez utilisé précédemment pour inspecter ses services et pods OpenShift. N'effectuez aucune modification sur le nœud, comme arrêter des services ou modifier des fichiers de configuration.
- 3.1. Démarrez une session shell sur le nœud, puis utilisez la commande `chroot` pour accéder au système de fichiers local de l'hôte.

```
[student@workstation ~]$ oc debug node/master01
Creating debug namespace/openshift-debug-node-5zsch ...
Starting pod/master01-debug ...
To use host binaries, run `chroot /host`
Pod IP: 192.168.50.10
If you do not see a command prompt, try pressing enter.
sh-4.4# chroot /host
sh-4.4#
```

- 3.2. Toujours en utilisant la même session shell, vérifiez que le Kubelet et le moteur de conteneur CRI-O sont en cours d'exécution. Saisissez `q` pour quitter la commande.

```
sh-4.4# systemctl status kubelet
● kubelet.service - Kubernetes Kubelet
  Loaded: loaded (/etc/systemd/system/kubelet.service; enabled; vendor preset:
  disabled)
  Drop-In: /etc/systemd/system/kubelet.service.d
            └─10-mco-default-madv.conf, 20-logging.conf, 20-nodenet.conf
  Active: active (running) since Wed 2022-04-20 20:36:05 UTC; 5h 47min ago
    ...output omitted...
q
```

Exécutez à nouveau la même commande sur le service `crio`. Saisissez `q` pour quitter la commande.

```
sh-4.4# systemctl status crio
● crio.service - Container Runtime Interface for OCI (CRI-O)
  Loaded: loaded (/usr/lib/systemd/system/crio.service; disabled; vendor preset:
  disabled)
  Drop-In: /etc/systemd/system/crio.service.d
            └─10-mco-default-madv.conf, 10-mco-profile-unix-socket.conf, 20-
  nodenet.conf
  Active: active (running) since Wed 2022-04-20 20:35:59 UTC; 5h 59min ago
    ...output omitted...
q
```

**chapitre 9 |** Vérification de la santé d'un cluster

- 3.3. Toujours en utilisant la même session shell, vérifiez que le pod `etcd` est en cours d'exécution.

```
sh-4.4# crictl ps --name etcd
CONTAINER      ... STATE      NAME          ATTEMPT  POD ID
5e18a8220a9d5  ... Running   etcd-operator  2        90b6b1150e3fe8d9adceeb5549
19bc3ed5e8643  ... Running   etcd-metrics   3        c94baa4f55618
...output omitted...
```

- 3.4. Mettez fin à la session `chroot` et à la session shell sur le nœud. Cette opération met également fin à la commande `oc debug node`.

```
sh-4.4# exit
exit

sh-4.4# exit
exit

Removing debug pod ...
[student@workstation ~]$
```

► 4. Accédez au projet `install-troubleshoot` pour diagnostiquer un pod en état d'erreur.

- 4.1. Utilisez le projet `install-troubleshoot`.

```
[student@workstation ~]$ oc project install-troubleshoot
Now using project "install-troubleshoot" on server ...
```

- 4.2. Vérifiez que le projet dispose d'un seul pod en état `ErrImagePull` ou `ImagePullBackOff`.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS            ...
pgsql-7d4cc9d6d-m5r59  0/1     ImagePullBackOff  ...
```

- 4.3. Vérifiez que le projet inclut un déploiement Kubernetes qui gère le pod.

```
[student@workstation ~]$ oc status
...output omitted...
svc/pgsql - 172.30.13.140:5432

deployment/pgsql deploys registry.redhat.io/rhel8/postgresq-13:1
  deployment #1 running for 7 hours - 0/1 pods
...output omitted...
```

- 4.4. Répertoriez tous les événements du projet en cours et recherchez les messages d'erreur liés au pod.

```
[student@workstation ~]$ oc get events
LAST SEEN    TYPE      REASON          OBJECT                  MESSAGE
7h17m       Normal    Scheduled       pod/pgsql-768d797946-d2gj4
  Successfully assigned install-troubleshoot/pgsql-768d797946-d2gj4 to master0
```

**chapitre 9 |** Vérification de la santé d'un cluster

```

2
7h17m      Normal   AddedInterface      pod/pgsql-768d797946-d2gj4   Add eth0
[10.9.0.21/23] from openshift-sdn
3h22m      Normal   Pulling            pod/pgsql-768d797946-d2gj4   Pulling
image "registry.redhat.io/rhel8/postgresq-13:1"
7h16m      Warning  Failed             pod/pgsql-768d797946-d2gj4   Failed to
pull image "registry.redhat.io/rhel8/postgresq-13:1": rpc error: code = Unknown
desc = reading manifest 1 in registry.redhat.io/rhel8/postgresq-13: unknown: Not
Found
7h16m      Warning  Failed             pod/pgsql-768d797946-d2gj4   Error:
ErrImagePull
2m32s      Normal   BackOff            pod/pgsql-768d797946-d2gj4   Back-off
pulling image "registry.redhat.io/rhel8/postgresq-13:1"
7h16m      Warning  Failed             pod/pgsql-768d797946-d2gj4   Error:
ImagePullBackOff
7h17m      Normal   SuccessfulCreate    replicaset/pgsql-768d797946   Created
pod: pgsql-768d797946-d2gj4
7h17m      Normal   ScalingReplicaSet  deployment/pgsql
replica set pgsql-768d797946 to 1

```

Cette sortie indique également un problème d'obtention de l'image pour le déploiement du pod.

- 4.5. Connectez-vous à Red Hat Container Catalog à l'aide de votre compte Red Hat.

```
[student@workstation ~]$ podman login registry.redhat.io
Username: your_username
Password: your_password
Login Succeeded!
```

- 4.6. Utilisez Skopeo pour trouver des informations sur l'image de conteneur à partir des événements.

```
[student@workstation ~]$ skopeo inspect \
> docker://registry.redhat.io/rhel8/postgresq-13:1
FATA[0000] Error parsing image name "docker://registry.redhat.io/rhel8/
postgresq-13:1": Error reading manifest 1 in registry.redhat.io/rhel8/
postgresq-13: unknown: Not Found
```

- 4.7. Il semble que l'image du conteneur soit mal orthographiée. Vérifiez qu'elle fonctionne si vous remplacez postgresq-13 par postgresql-13.

```
[student@workstation ~]$ skopeo inspect \
> docker://registry.redhat.io/rhel8/postgresql-13:1
{
  "Name": "registry.redhat.io/rhel8/postgresql-13",
  ...output omitted...
```

- 4.8. Pour vérifier que le nom de l'image est la cause de l'erreur, éditez le déploiement psql pour corriger le nom de l'image de conteneur. Les commandes oc edit utilisent vi comme éditeur par défaut.

**Mise en garde**

Dans un scénario réel, vous devez demander à la personne qui a déployé la base de données PostgreSQL de corriger son YAML et de redéployer son application.

```
[student@workstation ~]$ oc edit deployment/sql
...output omitted...
spec:
  containers:
    - env:
        - name: POSTGRESQL_DATABASE
          value: db
        - name: POSTGRESQL_PASSWORD
          value: pass
        - name: POSTGRESQL_USER
          value: user
      image: registry.redhat.io/rhel8/postgresql-13:1-7
...output omitted...
```

4.9. Vérifiez qu'un nouveau déploiement est actif.

```
[student@workstation ~]$ oc status
...output omitted...
deployment/sql deploys registry.redhat.io/rhel8/postgresql-13:1
  deployment #2 running for 30 seconds - 1 pod
  deployment #1 deployed 7 hours ago
```

4.10. Listez tous les pods du projet en cours. Vous verrez peut-être à la fois l'ancien pod en échec et le nouveau pod pendant quelques instants. Répétez la commande suivante jusqu'à ce que le nouveau pod soit prêt et en cours d'exécution, et que vous ne voyez plus l'ancien pod.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
sql-544c9c666f-btlw8  1/1     Running   0          55s
```

**Fin**

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab install-troubleshoot finish
```

L'exercice guidé est maintenant terminé.

# Présentation du stockage dynamique OpenShift

## Résultats

Après avoir terminé cette section, vous devez pouvoir identifier les composants et les ressources du stockage persistant, et déployer une application qui utilise une revendication de volume persistant.

## Présentation du stockage persistant

Par défaut, les conteneurs disposent d'un stockage éphémère. Par exemple, lorsqu'un conteneur est supprimé, tous les fichiers et toutes les données qu'il contient le sont également. Pour conserver les fichiers, les conteneurs offrent deux méthodes principales pour maintenir le stockage persistant : les volumes et les montages par liaison. Les volumes constituent la méthode OpenShift privilégiée pour la gestion du stockage persistant. Les volumes sont gérés manuellement par l'administrateur ou de manière dynamique par le biais d'une classe de stockage. Les développeurs qui utilisent des conteneurs sur un système local peuvent monter un répertoire local dans un conteneur à l'aide d'un montage par liaison.

Les administrateurs de cluster OpenShift utilisent la structure de volume persistant Kubernetes pour gérer le stockage persistant pour les utilisateurs d'un cluster. Il existe deux modes de déploiement du stockage pour le cluster : statique et dynamique. Dans le cas d'un déploiement statique, l'administrateur de cluster doit créer manuellement des volumes persistants. Le déploiement dynamique utilise des classes de stockage pour créer les volumes persistants à la demande.

OpenShift Container Platform utilise des classes de stockage pour permettre aux administrateurs de fournir un stockage persistant. Les classes de stockage constituent un moyen de décrire des types de stockage pour le cluster et de déployer le stockage dynamique à la demande.

Les développeurs utilisent des revendications de volume persistant pour ajouter des volumes persistants de manière dynamique à leurs applications ; il n'est pas nécessaire pour le développeur de connaître les détails de l'infrastructure de stockage. Avec le déploiement statique, les développeurs utilisent des volumes persistants (PV) existants ou demandent à un administrateur d'en créer manuellement pour leurs applications.

Une revendication de volume persistant (PVC) appartient à un projet spécifique. Pour créer une revendication de volume persistant, vous devez spécifier le mode d'accès et la taille, entre autres options. Une fois créée, une PVC ne peut pas être partagée entre des projets. Les développeurs utilisent une PVC pour accéder à un volume persistant (PV). Les volumes persistants ne sont pas exclusifs aux projets et sont accessibles sur l'ensemble du cluster OpenShift. Lorsqu'un volume persistant est lié à une revendication de volume persistant (PVC), il ne peut pas être lié à une autre.

## Cycle de vie des volumes persistants (PV) et des revendications de volume persistant (PVC)

Les revendications de volume persistant demandent des ressources de volumes persistants. Pour être éligible, un PV ne doit pas être lié à une autre PVC. En outre, le PV doit fournir le mode d'accès spécifié dans la PVC et doit avoir au moins la taille demandée dans la PVC. Une PVC peut spécifier des critères supplémentaires, tels que le nom d'une classe de stockage. Si une PVC ne

## chapitre 9 | Vérification de la santé d'un cluster

parvient pas à trouver un PV correspondant à tous les critères, elle passe à l'état Pending (En attente) et attend qu'un PV approprié soit disponible. Un administrateur de cluster peut créer le PV manuellement ou une classe de stockage peut le créer de manière dynamique. Un volume persistant lié peut être monté en tant que volume vers un point de montage spécifique dans le pod (par exemple, /var/lib/pgsql pour une base de données PostgreSQL).

## Vérification du stockage approvisionné dynamique

Utilisez la commande `oc get storageclass` pour afficher les classes de stockage disponibles. La sortie identifie la classe de stockage par défaut. S'il existe une classe de stockage, les volumes persistants sont créés de manière dynamique afin de correspondre aux revendications de volume persistant. Une revendication de volume persistant qui ne spécifie pas de classe de stockage utilise la classe de stockage par défaut.

```
[user@host ~]$ oc get storageclass
NAME          PROVISIONER
nfs-storage (default) nfs-storage-provisioner ...
```



### Note

L'environnement de formation utilise un approvisionneur NFS Open Source externe. Cet approvisionneur crée des volumes persistants NFS de manière dynamique à partir d'un serveur NFS existant. Red Hat déconseille d'utiliser cet approvisionneur dans des environnements de production.

## Déploiement d'un stockage approvisionné de manière dynamique

Pour ajouter un volume à une application, créez une ressource `PersistentVolumeClaim` et ajoutez-la à l'application en tant que volume. Créez la revendication de volume persistant à l'aide d'un manifeste Kubernetes ou de la commande `oc set volumes`. En plus de créer une revendication de volume persistant ou d'utiliser une revendication existante, la commande `oc set volumes` peut modifier un déploiement afin de monter la revendication de volume persistant en tant que volume dans le pod.

Pour ajouter un volume à une application, utilisez la commande `oc set volumes` :

```
[user@host ~]$ oc set volumes deployment/example-application \
>   --add --name example-pv-storage --type pvc --claim-class nfs-storage \
>   --claim-mode rwo --claim-size 15Gi --mount-path /var/lib/example-app \
>   --claim-name example-pv-claim
```

La commande crée une ressource de revendication de volume persistant et l'ajoute à l'application en tant que volume dans le pod.

L'exemple YAML suivant spécifie une revendication de volume persistant.

Pour créer un objet API `PersistentVolumeClaim`, procédez comme suit :

```
...
apiVersion: v1
kind: PersistentVolumeClaim ①
metadata:
```

```

name: example-pv-claim ②
labels:
  app: example-application
spec:
  accessModes:
    - ReadWriteOnce ③
  resources:
    requests:
      storage: 15Gi ④

```

- ① Indique qu'il s'agit d'une revendication de volume persistant.
- ② Nom à utiliser dans le champ `claimName` de l'élément `persistentVolumeClaim` dans la section `volumes` d'un manifeste de déploiement.
- ③ L'approvisionneur de classe de stockage doit fournir ce mode d'accès. Si des volumes persistants sont créés de manière statique, un volume persistant éligible doit alors fournir ce mode d'accès.
- ④ La classe de stockage va créer un volume persistant correspondant à cette demande de taille. Si des volumes persistants sont créés de manière statique, un volume persistant éligible doit avoir au moins la taille demandée.

OpenShift définit trois modes d'accès qui sont récapitulés dans le tableau ci-dessous.

Mode d'accès	Abréviation de l'interface de ligne de commande	Description
ReadWriteMany	RWX	Kubernetes peut monter le volume en lecture/écriture sur de nombreux nœuds.
ReadOnlyMany	ROX	Kubernetes peut monter le volume en lecture seule sur de nombreux nœuds.
ReadWriteOnce	RWO	Kubernetes peut monter le volume en lecture/écriture sur un seul nœud.

Il est important de mentionner que les revendications correspondent au meilleur PV disponible, généralement avec un mode d'accès similaire, mais les modes pris en charge dépendent des fonctionnalités du fournisseur. Par exemple, vous pouvez avoir une PVC avec RWO demandant un PV NFS, et elle peut être comparée, car NFS prend en charge RWO, mais cela ne peut pas arriver en sens inverse et la requête gardera l'état pending (en attente).

Pour ajouter la PVC à l'application :

```

...output omitted...
spec:
  volumes:
    - name: example-pv-storage
      persistentVolumeClaim:
        claimName: example-pv-claim
  containers:
    - name: example-application
      image: registry.redhat.io/rhel8/example-app

```

```
ports:  
- containerPort: 1234  
volumeMounts:  
- mountPath: "/var/lib/example-app"  
  name: example-pv-storage  
...output omitted...
```

## Suppression de revendications de volume persistant

Pour supprimer un volume, utilisez la commande `oc delete` afin de supprimer la revendication de volume persistant. La classe de stockage récupère le volume après la suppression de la PVC.

```
[user@host ~]$ oc delete pvc/example-pvc-storage
```



### Références

Pour plus d'informations sur le stockage persistant, reportez-vous au chapitre *Understanding persistent storage* de la documentation *Storage* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/storage/index#understanding-persistent-storage](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/storage/index#understanding-persistent-storage)

Pour plus d'informations sur le stockage éphémère, reportez-vous au chapitre *Understanding ephemeral storage* de la documentation *Storage* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/storage/index#understanding-ephemeral-storage](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/storage/index#understanding-ephemeral-storage)

## ► Exercice guidé

# Présentation du stockage dynamique OpenShift

Dans cet exercice, vous allez déployer une base de données PostgreSQL à l'aide d'une revendication de volume persistant et identifier le volume qui lui est alloué de manière dynamique.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Identifier les paramètres de stockage par défaut d'un cluster OpenShift.
- Créer des revendications de volume persistant (PVC).
- Gérer des volumes persistants.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande garantit que l'API du cluster est accessible et télécharge les fichiers nécessaires à cet exercice.

```
[student@workstation ~]$ lab install-storage start
```

## Instructions

► 1. Connectez-vous au cluster OpenShift.

1. Procurez-vous le fichier de configuration de la salle de classe, accessible à l'adresse `/usr/local/etc/ocp4.config`.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

2. Connectez-vous au cluster en tant qu'utilisateur `kubeadmin`. Si vous y êtes invité, acceptez le certificat non sécurisé.

```
[student@workstation ~]$ oc login -u kubeadmin -p ${RHT_OCP4_KUBEADM_PASSWD} \
>   https://api.ocp4.example.com:6443
...output omitted...
```

► 2. Créez un nouveau projet nommé `install-storage`.

**chapitre 9 |** Vérification de la santé d'un cluster

```
[student@workstation ~]$ oc new-project install-storage
Now using project "install-storage" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 3. Vérifiez la classe de stockage par défaut.

```
[student@workstation ~]$ oc get storageclass
NAME           PROVISIONER      ...
nfs-storage   (default)   k8s-sigs.io/nfs-subdir-external-provisioner ...
```

- 4. Créez un déploiement de base de données à l'aide de l'image de conteneur située à l'emplacement suivant: `registry.redhat.io/rhel8/postgresql-13:1-7`.

```
[student@workstation ~]$ oc new-app --name postgresql-persistent \
>   --image registry.redhat.io/rhel8/postgresql-13:1-7 \
>   -e POSTGRESQL_USER=redhat \
>   -e POSTGRESQL_PASSWORD=redhat123 \
>   -e POSTGRESQL_DATABASE=persistentdb
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "postgresql-persistent" created
deployment.apps "postgresql-persistent" created
service "postgresql-persistent" created
--> Success
...output omitted...
```

**Note**

Pour plus de commodité, le fichier `~/D0280/labs/install-storage/commands.txt` contient des commandes que vous pouvez copier et coller.

- 5. Ajoutez un volume persistant pour la base de données PostgreSQL.

- 5.1. Créez une revendication de volume persistant pour ajouter un nouveau volume au déploiement `postgresql-persistent`.

```
[student@workstation ~]$ oc set volumes deployment/postgresql-persistent \
>   --add --name postgresql-storage --type pvc --claim-class nfs-storage \
>   --claim-mode rwo --claim-size 10Gi --mount-path /var/lib/pgsql \
>   --claim-name postgresql-storage
deployment.apps/postgresql-persistent volume updated
```

- 5.2. Vérifiez que la PVC a bien été créée.

```
[student@workstation ~]$ oc get pvc
NAME          STATUS    ...  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
postgresql-storage  Bound    ...  10Gi       RWO          nfs-storage  25s
```

- 5.3. Vérifiez que le PV a bien été créé.

```
[student@workstation ~]$ oc get pv \
> -o custom-columns=NAME:.metadata.name,CLAIM:.spec.claimRef.name
NAME                                CLAIM
pvc-26cc804a-4ec2-4f52-b6e5-84404b4b9def   image-registry-storage
pvc-65c3cce7-45eb-482d-badf-a6469640bd75   postgresql-storage
```

- 6. Renseignez la base de données à l'aide du script ~/D0280/labs/install-storage/init\_data.sh.

6.1. Exécutez le script init\_data.sh.

```
[student@workstation ~]$ cd ~/D0280/labs/install-storage
[student@workstation install-storage]$ ./init_data.sh
Populating characters table
CREATE TABLE
INSERT 0 5
```

6.2. Utilisez le script ~/D0280/labs/install-storage/check\_data.sh pour vérifier que la base de données a bien été remplie.

```
[student@workstation install-storage]$ ./check_data.sh
Checking characters table
+---+-----+
| id |      name      |      nationality |
+---+-----+
| 1 | Wolfgang Amadeus Mozart | Prince-Archbishopric of Salzburg |
| 2 | Ludwig van Beethoven | Bonn, Germany |
| 3 | Johann Sebastian Bach | Eisenach, Germany |
| 4 | José Pablo Moncayo | Guadalajara, México |
| 5 | Niccolò Paganini | Genoa, Italy |
(5 rows)
```

- 7. Supprimez le déploiement postgresql-persistent et créez-en un autre nommé postgresql-deployment2 qui utilise le même volume persistant ; vérifiez que les données ont été conservées.

7.1. Supprimez toutes les ressources qui contiennent le libellé app=postgresql-persistent.

```
[student@workstation install-storage]$ oc delete all -l app=postgresql-persistent
service "postgresql-persistent" deleted
deployment.apps "postgresql-persistent" deleted
imagestream.image.openshift.io "postgresql-persistent" deleted
```

7.2. Créez le déploiement postgresql-persistent2 avec les mêmes données d'initialisation que le déploiement postgresql-persistent.

```
[student@workstation install-storage]$ oc new-app --name postgresql-persistent2 \
> --image registry.redhat.io/rhel8/postgresql-13:1-7 \
> -e POSTGRESQL_USER=redhat \
> -e POSTGRESQL_PASSWORD=redhat123 \
> -e POSTGRESQL_DATABASE=persistentdb
```

**chapitre 9 |** Vérification de la santé d'un cluster

```
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "postgresql-persistent2" created
deployment.apps "postgresql-persistent2" created
service "postgresql-persistent2" created
--> Success
...output omitted...
```

- 7.3. Utilisez le script `~/D0280/labs/install-storage/check_data.sh` pour vérifier que la base de données ne contient pas la table de caractères.

```
[student@workstation install-storage]$ ./check_data.sh
Checking characters table
ERROR: 'characters' table does not exist
```

- 7.4. Ajoutez la revendication de volume persistant `postgresql-storage` au déploiement `postgresql-persistent2`.

```
[student@workstation install-storage]$ oc set volumes \
>   deployment/postgresql-persistent2 \
>     --add --name postgresql-storage --type pvc \
>     --claim-name postgresql-storage --mount-path /var/lib/pgsql
deployment.apps/postgresql-persistent2 volume updated
```

- 7.5. Utilisez le script `~/D0280/labs/install-storage/check_data.sh` pour vérifier que le volume persistant a bien été ajouté et que le pod `postgresql-persistent2` peut accéder aux données créées précédemment.

```
[student@workstation install-storage]$ ./check_data.sh
Checking characters table
+-----+
| id | name | nationality |
+-----+
| 1 | Wolfgang Amadeus Mozart | Prince-Archbishopric of Salzburg |
| 2 | Ludwig van Beethoven | Bonn, Germany |
...output omitted...
```

- 8. Supprimez le déploiement `postgresql-persistent2` et la revendication de volume persistant.

- 8.1. Supprimez toutes les ressources qui contiennent le libellé `app=postgresql-persistent2`.

```
[student@workstation install-storage]$ oc delete all -l app=postgresql-persistent2
service "postgresql-persistent2" deleted
deployment.apps "postgresql-persistent2" deleted
imagestream.image.openshift.io "postgresql-persistent2" deleted
```

- 8.2. Supprimez le volume persistant en supprimant la revendication de volume persistant `postgresql-storage`, puis revenez au répertoire personnel.

```
[student@workstation install-storage]$ oc delete pvc/postgresql-storage  
persistentvolumeclaim "postgresql-storage" deleted
```

8.3. Revenez au répertoire /home/student.

```
[student@workstation install-storage]$ cd ~
```

## Fin

Sur la machine **workstation**, utilisez la commande **lab** pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab install-storage finish
```

L'exercice guidé est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Red Hat OpenShift Container Platform propose deux méthodes d'installation principales : l'automatisation de la pile complète et l'infrastructure préexistante.
- Les prochaines versions devraient intégrer d'autres fournisseurs de cloud et de virtualisation, tels que VMware, Red Hat Virtualization et IBM System Z.
- Un nœud OpenShift basé sur Red Hat Enterprise Linux CoreOS exécute très peu de services locaux qui nécessiteraient un accès direct à un nœud pour inspecter leur état. La plupart des services système s'exécutent en tant que conteneurs, à l'exception du moteur de conteneur CRI-O et du Kubelet.
- Les commandes `oc get node`, `oc adm top`, `oc adm node-logs` et `oc debug` fournissent des informations de résolution des problèmes sur les nœuds OpenShift.

## chapitre 10

# Configuration de l'authentification et de l'autorisation

### Objectif

Configurer l'authentification avec le fournisseur d'identité HTPasswd, et attribuer des rôles à des utilisateurs et des groupes.

### Résultats

- Configurer le fournisseur d'identité HTPasswd pour l'authentification OpenShift.
- Définir les contrôles d'accès basés sur les rôles et appliquer des permissions aux utilisateurs.

### Sections

- Configuration des fournisseurs d'identité (et exercice guidé)
- Définition et application d'autorisations à l'aide des RBAC (et exercice guidé)

### Atelier

Vérification de la santé d'un cluster

# Configuration des fournisseurs d'identité

---

## Résultats

À la fin de cette section, vous devez pouvoir configurer le fournisseur d'identité HTPasswd pour l'authentification OpenShift.

## Description des utilisateurs et des groupes OpenShift

Il existe plusieurs ressources OpenShift liées à l'authentification et à l'autorisation. Voici la liste des principaux types de ressources, accompagnés de leurs définitions :

### User

Dans l'architecture d'OpenShift Container Platform, les utilisateurs sont des entités qui interagissent avec le serveur d'API. La ressource User représente un acteur au sein du système. Attribuez des autorisations en ajoutant des rôles directement à l'utilisateur ou aux groupes dont l'utilisateur est membre.

### Identity

La ressource Identity conserve un enregistrement des tentatives d'authentification réussies d'un utilisateur et d'un fournisseur d'identité spécifiques. Toutes les données relatives à la source de l'authentification sont stockées sur l'identité. Seule une ressource User unique est associée à une ressource Identity.

### Service Account (Compte de service)

Dans OpenShift, les applications peuvent communiquer de manière indépendante avec l'API lorsque les informations d'identification de l'utilisateur ne peuvent pas être acquises. Pour préserver l'intégrité des informations d'identification d'un utilisateur standard, celles-ci ne sont pas partagées et les comptes de service sont utilisés à la place. Les comptes de service vous permettent de contrôler l'accès aux API sans avoir à emprunter les informations d'identification d'un utilisateur standard.

### Group

Les groupes représentent un ensemble spécifique d'utilisateurs. Les utilisateurs sont affectés à un ou plusieurs groupes. Les groupes sont mis à profit lors de la mise en œuvre des politiques d'autorisation pour attribuer des autorisations à plusieurs utilisateurs en même temps. Par exemple, si vous souhaitez autoriser vingt utilisateurs à accéder aux objets d'un projet, il est avantageux d'utiliser un groupe plutôt que d'accorder l'accès à chacun des utilisateurs individuellement. OpenShift Container Platform fournit également des groupes système ou groupes virtuels qui sont déployés automatiquement par le cluster.

### Rôle

Un rôle définit un ensemble d'autorisations permettant à un utilisateur d'effectuer des opérations API sur un ou plusieurs types de ressources. Vous accordez des autorisations à des utilisateurs, des groupes et des comptes de service en leur affectant des rôles.

Les ressources User et Identity ne sont généralement pas créées à l'avance. Elles sont généralement créées automatiquement par OpenShift après une connexion interactive réussie à l'aide d'OAuth.

## Authentification des demandes d'API

L'autorisation et l'authentification sont les deux couches de sécurité qui permettent à un utilisateur d'interagir avec le cluster. Lorsqu'un utilisateur adresse une demande à l'API, celle-ci l'associe à la demande. La couche d'authentification authentifie l'utilisateur. Une fois l'authentification effectuée, la couche d'autorisation décide d'accepter ou de rejeter la demande de l'API. La couche d'autorisation utilise des politiques de contrôle d'accès basé sur les rôles (RBAC) pour déterminer les priviléges des utilisateurs.

L'API OpenShift dispose de deux méthodes pour authentifier les demandes :

- les jetons d'accès OAuth ;
- les certificats clients X.509.

Si la demande ne comporte pas de jeton d'accès ou de certificat, la couche d'authentification lui attribue l'utilisateur virtuel `system:anonymous` et le groupe virtuel `system:unauthenticated`.

## Présentation de l'opérateur d'authentification

OpenShift Container Platform fournit l'opérateur d'authentification, qui exécute un serveur OAuth. Le serveur OAuth fournit des jetons d'accès OAuth aux utilisateurs lorsqu'ils tentent de s'authentifier auprès de l'API. Un fournisseur d'identité doit être configuré et disponible pour le serveur OAuth. Le serveur OAuth utilise un fournisseur d'identité pour valider l'identité du demandeur. Le serveur concilie l'utilisateur avec l'identité et crée le jeton d'accès OAuth pour l'utilisateur. OpenShift crée automatiquement des ressources Identity et User après une connexion réussie.

## Présentation des fournisseurs d'identité

Le serveur OAuth d'OpenShift peut être configuré de manière à utiliser de nombreux fournisseurs d'identité. La liste ci-dessous présente les fournisseurs les plus courants :

### **HTPasswd**

Valide les noms d'utilisateur et les mots de passe en fonction d'un secret qui stocke les informations d'identification générées à l'aide de la commande `htpasswd`.

### **de Keystone**

Permet d'activer l'authentification partagée avec un serveur OpenStack Keystone v3.

### **LDAP**

Configure le fournisseur d'identité LDAP pour valider les noms d'utilisateur et les mots de passe en fonction d'un serveur LDAPv3, à l'aide de l'authentification de liaison simple.

### **GitHub ou GitHub Enterprise**

Configure un fournisseur d'identité GitHub pour valider les noms d'utilisateur et les mots de passe en fonction d'un serveur d'authentification OAuth GitHub ou GitHub Enterprise.

### **OpenID Connect**

Permet d'intégrer un fournisseur d'identité OpenID Connect à l'aide d'un flux de code d'autorisation.

Vous devez mettre à jour la ressource personnalisée OAuth avec le fournisseur d'identité souhaité. Vous pouvez définir plusieurs fournisseurs d'identité, de types identiques ou différents, sur la même ressource personnalisée OAuth.

## Authentification en tant qu'administrateur de cluster

Avant de pouvoir configurer un fournisseur d'identité et de gérer des utilisateurs, vous devez accéder à votre cluster OpenShift en tant qu'administrateur de cluster. Un cluster OpenShift nouvellement installé offre deux moyens d'authentifier les demandes d'API avec les privilèges d'administrateur de cluster :

- Utilisez le fichier `kubeconfig` qui comprend un certificat client X.509 qui n'expire jamais.
- Authentifiez-vous en tant qu'utilisateur virtuel `kubeadmin`. Une authentification réussie accorde un token d'accès OAuth.

Pour créer des utilisateurs supplémentaires et leur accorder des niveaux d'accès différents, vous devez configurer un fournisseur d'identité et assigner des rôles à vos utilisateurs.

### Authentification à l'aide du certificat X.509

Au cours de l'installation, le programme d'installation d'OpenShift crée un fichier `kubeconfig` unique dans le répertoire `auth`. Le fichier `kubeconfig` contient des informations et des paramètres spécifiques utilisés par l'interface de ligne de commande pour connecter un client au serveur d'API approprié, y compris un certificat X.509.

Les journaux d'installation indiquent l'emplacement du fichier `kubeconfig` :

```
INFO Run 'export KUBECONFIG=root/auth/kubeconfig' to manage the cluster with 'oc'.
```



#### Note

Dans l'environnement de formation, la machine `utility` héberge le fichier `kubeconfig` sous `/home/lab/ocp4/auth/kubeconfig`.

Pour utiliser le fichier `kubeconfig` afin d'authentifier les commandes `oc`, vous devez le copier sur votre station de travail et définir le chemin absolu ou relatif vers la variable d'environnement `KUBECONFIG`. Vous pouvez ensuite exécuter toute commande `oc` qui nécessite des privilèges d'administrateur de cluster sans vous connecter à OpenShift.

```
[user@host ~]$ export KUBECONFIG=/home/user/auth/kubeconfig
[user@host ~]$ oc get nodes
```

Vous pouvez également utiliser l'option `--kubeconfig` de la commande `oc`.

```
[user@host ~]$ oc --kubeconfig /home/user/auth/kubeconfig get nodes
```

### Authentification à l'aide de l'utilisateur virtuel `kubeadmin`

Une fois l'installation terminée, OpenShift crée l'utilisateur virtuel `kubeadmin`. Le secret `kubeadmin` dans l'espace de noms `kube-system` contient le mot de passe haché pour l'utilisateur `kubeadmin`. L'utilisateur `kubeadmin` dispose des privilèges d'administrateur du cluster.

Le programme d'installation d'OpenShift génère de manière dynamique un mot de passe `kubeadmin` unique pour le cluster. Les journaux d'installation fournissent les informations

d'identification kubeadmin permettant de se connecter au cluster. Les journaux d'installation du cluster indiquent également la connexion, le mot de passe et l'URL de l'accès à la console.

```
...output omitted...
INFO The cluster is ready when 'oc login -u kubeadmin -p shdu_trbi_6ucX_edbu_aqop'
...output omitted...
INFO Access the OpenShift web-console here:
https://console-openshift-console.apps.ocp4.example.com
INFO Login to the console with user: kubeadmin, password: shdu_trbi_6ucX_edbu_aqop
```



### Note

Dans l'environnement de formation, la machine utility stocke le mot de passe de l'utilisateur kubeadmin dans le fichier /home/lab/ocp4/auth/kubeadmin-password.

## Suppression de l'utilisateur virtuel

Après avoir défini un fournisseur d'identité, créé un utilisateur et attribué à cet utilisateur le rôle `cluster-admin`, vous pouvez supprimer les informations d'identification de l'utilisateur `kubeadmin` pour améliorer la sécurité du cluster.

```
[user@host ~]$ oc delete secret kubeadmin -n kube-system
```



### Mise en garde

Si vous supprimez le secret `kubeadmin` avant de configurer un autre utilisateur avec des priviléges d'administrateur de cluster, vous ne pourrez administrer votre cluster qu'en utilisant le fichier `kubeconfig`. Si vous n'avez pas conservé de copie de ce fichier en lieu sûr, il n'y a aucun moyen de récupérer l'accès administratif à votre cluster. La seule alternative consiste à détruire et à réinstaller votre cluster.



### Mise en garde

Ne supprimez l'utilisateur `kubeadmin` à **aucun** moment pendant ce cours. L'utilisateur `kubeadmin` est essentiel à l'architecture des exercices pratiques du cours. La suppression de l'utilisateur `kubeadmin` nuit à l'environnement des exercices pratiques, ce qui vous oblige à créer un nouvel environnement d'exercices pratiques.

## Configuration du fournisseur d'identité HTPasswd

Le fournisseur d'identité HTPasswd valide les utilisateurs en fonction d'un secret qui contient les noms d'utilisateur et les mots de passe générés à l'aide de la commande `htpasswd` du projet du serveur HTTP Apache. Seul un administrateur de cluster peut modifier les données à l'intérieur du secret HTPasswd. Les utilisateurs standard ne peuvent pas modifier leurs propres mots de passe.

La gestion des utilisateurs à l'aide du fournisseur d'identité HTPasswd peut suffire pour un environnement de preuve de concept comportant un petit ensemble d'utilisateurs. Cependant, la plupart des environnements de production nécessitent un fournisseur d'identité plus puissant qui s'intègre au système de gestion des identités de l'organisation.

## Configuration de la ressource personnalisée OAuth

Pour utiliser le fournisseur d'identité HTPasswd, vous devez modifier la ressource personnalisée OAuth afin d'ajouter une entrée à la matrice `.spec.identityProviders`:

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  name: cluster
spec:
  identityProviders:
    - name: my_htpasswd_provider ①
      mappingMethod: claim ②
      type: HTPasswd
      htpasswd:
        fileData:
          name: htpasswd-secret ③
```

- ① Ce nom de fournisseur est précédé des noms d'utilisateur du fournisseur pour former un nom d'identité.
- ② Contrôle la manière dont les mappages sont établis entre les identités du fournisseur et les objets utilisateur.
- ③ Secret existant contenant des données générées à l'aide de la commande `htpasswd`.

## Mise à jour de la ressource personnalisée OAuth

Pour mettre à jour la ressource personnalisée OAuth, utilisez la commande `oc get` pour exporter la ressource de cluster OAuth existante vers un fichier au format YAML.

```
[user@host ~]$ oc get oauth cluster -o yaml > oauth.yaml
```

Ouvrez ensuite le fichier obtenu dans un éditeur de texte et apportez les modifications nécessaires aux paramètres intégrés du fournisseur d'identité.

Une fois les modifications terminées et le fichier enregistré, vous devez appliquer la nouvelle ressource personnalisée à l'aide de la commande `oc replace`.

```
[user@host ~]$ oc replace -f oauth.yaml
```

## Gestion des utilisateurs à l'aide du fournisseur d'identité HTPasswd

La gestion des informations d'identification de l'utilisateur avec le fournisseur d'identité HTPasswd nécessite de créer un fichier `htpasswd` temporaire, d'apporter des modifications au fichier et d'appliquer ces modifications au secret.

### Création d'un fichier HTPasswd

Le paquetage `httpd-tools` fournit l'utilitaire `htpasswd`, doit être installé et disponible sur votre système.

Créez le fichier `htpasswd`.

```
[user@host ~]$ htpasswd -c -B -b /tmp/htpasswd student redhat123
```

**Important**

N'utilisez l'option `-c` que lors de la création d'un fichier. L'option `-c` remplace tout le contenu du fichier si le fichier existe déjà.

Ajoutez ou mettez à jour les informations d'identification.

```
[user@host ~]$ htpasswd -B -b /tmp/htpasswd student redhat1234
```

Supprimez les informations d'identification.

```
[user@host ~]$ htpasswd -D /tmp/htpasswd student
```

## Création du secret HTPasswd

Pour utiliser le fournisseur HTPasswd, vous devez créer un secret contenant les données du fichier `htpasswd`. Dans l'exemple suivant, le secret est nommé `htpasswd-secret`.

```
[user@host ~]$ oc create secret generic htpasswd-secret \
>   --from-file htpasswd=/tmp/htpasswd -n openshift-config
```

**Important**

Un secret utilisé par le fournisseur d'identité HTPasswd nécessite l'ajout du préfixe `htpasswd=` avant de spécifier le chemin d'accès au fichier.

## Extraction des données du secret

Lors de l'ajout ou de la suppression d'utilisateurs, un administrateur ne peut pas supposer qu'un fichier `htpasswd` local est valide. De plus, l'administrateur n'est peut-être pas sur un système disposant du fichier `htpasswd`. En situation réelle, il incomberait à l'administrateur d'utiliser la commande `oc extract`.

Par défaut, la commande `oc extract` enregistre chaque clé dans un mappage de configuration ou un secret sous la forme d'un fichier distinct. Sinon, toutes les données peuvent ensuite être redirigées vers un fichier ou affichées sous forme de sortie standard. Pour extraire les données du secret `htpasswd-secret` et les rediriger vers un nom de fichier `/tmp/`, utilisez la commande suivante : L'option `--confirm` remplace tout le contenu du fichier si le fichier existe déjà.

```
[user@host ~]$ oc extract secret/htpasswd-secret -n openshift-config \
>   --to /tmp/ --confirm /tmp/htpasswd
```

## Mise à jour du secret HTPasswd

Le secret doit être mis à jour après l'ajout, la modification ou la suppression d'utilisateurs. Utilisez la commande `oc set data secret` pour mettre à jour un secret. À moins que le nom du fichier ne soit `htpasswd`, vous devez spécifier `htpasswd=` pour mettre à jour la clé `htpasswd` dans le secret.

## chapitre 10 | Configuration de l'authentification et de l'autorisation

La commande suivante met à jour le secret `htpasswd-secret` dans l'espace de noms `openshift-config` à l'aide du contenu du fichier `/tmp/htpasswd`.

```
[user@host ~]$ oc set data secret/htpasswd-secret \
>   --from-file htpasswd=/tmp/htpasswd -n openshift-config
```

Après la mise à jour du secret, l'opérateur OAuth redéploie les pods dans l'espace de noms `openshift-authentication`. Surveillez le redéploiement des nouveaux pods OAuth en exécutant :

```
[user@host ~]$ watch oc get pods -n openshift-authentication
```

Testez les ajouts, modifications ou suppressions apportés au secret après la fin du déploiement des nouveaux pods.

## Suppression des utilisateurs et des identités

Lorsqu'un scénario nécessite la suppression d'un utilisateur, il ne suffit pas de supprimer l'utilisateur du fournisseur d'identité. Les ressources User et Identity doivent également être supprimées.

Vous devez supprimer le mot de passe du secret `htpasswd`, supprimer l'utilisateur du fichier `htpasswd` local, puis mettre à jour le secret.

Pour supprimer l'utilisateur de `htpasswd`, exécutez la commande suivante :

```
[user@host ~]$ htpasswd -D /tmp/htpasswd manager
```

Mettez à jour le secret pour supprimer tout ce qui reste du mot de passe de l'utilisateur.

```
[user@host ~]$ oc set data secret/htpasswd-secret \
>   --from-file htpasswd=/tmp/htpasswd -n openshift-config
```

Supprimez la ressource User à l'aide de la commande suivante :

```
[user@host ~]$ oc delete user manager
user.user.openshift.io "manager" deleted
```

Les ressources d'identité incluent le nom du fournisseur d'identités. Pour supprimer la ressource d'identité de l'utilisateur `manager`, recherchez la ressource, puis supprimez-la.

```
[user@host ~]$ oc get identities | grep manager
my_htpasswd_provider:manager    my_htpasswd_provider    manager        manager    ...
[user@host ~]$ oc delete identity my_htpasswd_provider:manager
identity.user.openshift.io "my_htpasswd_provider:manager" deleted
```

## Attribution de privilèges administratifs

Le rôle `cluster-admin` à l'échelle du cluster accorde des privilèges d'administration de cluster aux utilisateurs et aux groupes. Il permet à l'utilisateur d'effectuer n'importe quelle action sur

toutes les ressources au sein du cluster. Dans l'exemple suivant, le rôle `cluster-admin` est affecté à l'utilisateur `student`.

```
[user@host ~]$ oc adm policy add-cluster-role-to-user cluster-admin student
```



## Références

Pour plus d'informations sur les fournisseurs d'identité, reportez-vous au chapitre Understanding identity provider configuration de la documentation Authentication and authorization de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/authentication\\_and\\_authorization/index#understanding-identity-provider](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/authentication_and_authorization/index#understanding-identity-provider)

## ► Exercice guidé

# Configuration des fournisseurs d'identité

Dans cet exercice, vous allez configurer le fournisseur d'identité HTPasswd et créer des utilisateurs pour les administrateurs de cluster.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Créer des utilisateurs et des mots de passe pour l'authentification HTPasswd.
- Configurer le fournisseur d'identité pour l'authentification HTPasswd.
- Affecter des droits d'administration de cluster aux utilisateurs.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

```
[student@workstation ~]$ lab auth-provider start
```

Cette commande garantit que l'API du cluster est accessible, que le paquetage `httpd-utils` est installé et que les paramètres d'authentification sont configurés selon les valeurs d'installation par défaut.

## Instructions

- 1. Ajoutez une entrée pour deux utilisateurs `htpasswd`, `admin` et `developer`. Affectez à l'utilisateur `admin` le mot de passe `redhat` et à l'utilisateur `developer` le mot de passe `developer`.
- 1.1. Procurez-vous le fichier de configuration de la salle de classe, accessible à l'adresse `/usr/local/etc/ocp4.config`.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```



### Mise en garde

Ne supprimez l'utilisateur `kubeadmin` à **aucun** moment pendant ce cours. L'utilisateur `kubeadmin` est essentiel à l'architecture des exercices pratiques du cours. La suppression de l'utilisateur `kubeadmin` nuit à l'environnement des exercices pratiques, ce qui vous oblige à créer un nouvel environnement d'exercices pratiques.

- 1.2. Créez un fichier d'authentification HTPasswd nommé `htpasswd` dans le répertoire `~/D0280/labs/auth-provider/`. Ajoutez l'utilisateur `admin` avec le mot de passe `redhat`. Le nom du fichier est arbitraire, mais cet exercice utilise le fichier `~/D0280/labs/auth-provider/htpasswd`.

Utilisez la commande `htpasswd` pour remplir le fichier d'authentification HTPasswd avec les noms d'utilisateur et les mots de passe chiffrés. L'option `-B` utilise le chiffrage bcrypt. Par défaut, la commande `htpasswd` utilise l'algorithme de hachage MD5 lorsque vous ne spécifiez pas d'autre algorithme.

```
[student@workstation ~]$ htpasswd -c -B -b ~/D0280/labs/auth-provider/htpasswd \
> admin redhat
Adding password for user admin
```

- 1.3. Ajoutez l'utilisateur `developer` avec le mot de passe `developer` au fichier `~/D0280/labs/auth-provider/htpasswd`. Le mot de passe de l'utilisateur `developer` est stocké à l'aide de MD5, car aucun algorithme de hachage n'a été spécifié lors de l'appel de la commande `htpasswd`.

```
[student@workstation ~]$ htpasswd -b ~/D0280/labs/auth-provider/htpasswd \
> developer developer
Adding password for user developer
```

- 1.4. Examinez le contenu de votre fichier `~/D0280/labs/auth-provider/htpasswd` et vérifiez qu'il contient deux entrées avec des mots de passe hachés : une pour l'utilisateur `admin` et une autre pour l'utilisateur `developer`.

```
[student@workstation ~]$ cat ~/D0280/labs/auth-provider/htpasswd
admin:$2y$05$QPuzHdI06IDkJsST.tdkZuSmgjUHV1XeYU4FjxhQrFqKL7hs2ZUl6
developer:$apr1$0Nzmc1rh$yGtne1k.JX6L5s5wNa2ye.
```

- ▶ 2. Connectez-vous à OpenShift et créez un secret contenant le fichier des utilisateurs HTPasswd.

- 2.1. Connectez-vous au cluster en tant qu'utilisateur `kubeadmin`.

```
[student@workstation ~]$ oc login -u kubeadmin -p ${RHT_OCP4_KUBEADM_PASSWD} \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 2.2. Créez un secret à partir du fichier `~/D0280/labs/auth-provider/htpasswd`. Pour utiliser le fournisseur d'identité HTPasswd, vous devez définir un secret avec une clé nommée `htpasswd` qui contient le fichier des utilisateurs HTPasswd `~/D0280/labs/auth-provider/htpasswd`.



### Important

Un secret utilisé par le fournisseur d'identité HTPasswd nécessite l'ajout du préfixe `htpasswd=` avant de spécifier le chemin d'accès au fichier.

```
[student@workstation ~]$ oc create secret generic localusers \
>   --from-file htpasswd=~/D0280/labs/auth-provider/htpasswd \
>   -n openshift-config
secret/localusers created
```

2.3. Assignez le rôle `admin` à l'utilisateur `cluster-admin`.

```
[student@workstation ~]$ oc adm policy add-cluster-role-to-user \
>   cluster-admin admin
Warning: User 'admin' not found
clusterrole.rbac.authorization.k8s.io/cluster-admin added: "admin"
```



#### Note

Le résultat indique que l'utilisateur `admin` est introuvable et peut être ignoré sans risque.

- 3. Mettez à jour le fournisseur d'identité HTPasswd du cluster afin que vos utilisateurs puissent s'authentifier. Configurez le fichier de ressources personnalisées et mettez à jour le cluster.

3.1. Exportez la ressource OAuth existante vers un fichier nommé `oauth.yaml` dans le répertoire `~/D0280/labs/auth-provider`.

```
[student@workstation ~]$ oc get oauth cluster \
>   -o yaml > ~/D0280/labs/auth-provider/oauth.yaml
```



#### Note

Un fichier `oauth.yaml` contenant le fichier de ressources personnalisées rempli est téléchargé pour votre commodité dans `~/D0280/solutions/auth-provider`.

- 3.2. Modifiez le fichier `~/D0280/labs/auth-provider/oauth.yaml` à l'aide de l'éditeur de texte de votre choix. Vous pouvez choisir le nom des structures `identityProviders` et `fileData`. Pour cet exercice, utilisez respectivement les valeurs `myusers` et `localusers`.

La ressource personnalisée remplie doit correspondre à ce qui suit. Notez que `htpasswd`, `mappingMethod`, `name` et `type` se trouvent au même niveau de mise en retrait.

```
apiVersion: config.openshift.io/v1
kind: OAuth
...output omitted...
spec:
  identityProviders:
    - htpasswd:
        fileData:
          name: localusers
```

```
mappingMethod: claim
name: myusers
type: HTPasswd
```

- 3.3. Appliquez la ressource personnalisée définie à l'étape précédente.

```
[student@workstation ~]$ oc replace -f ~/DO280/labs/auth-provider/oauth.yaml
oauth.config.openshift.io/cluster replaced
```



#### Note

Les pods de l'espace de noms openshift-authentication seront redéployés si la commande `oc replace` réussit. Si le secret a bien été créé précédemment, vous pouvez vous connecter à l'aide du fournisseur d'identité HTPasswd.

- ▶ 4. Connectez-vous en tant qu'utilisateur `admin` et `developer` pour vérifier la configuration utilisateur HTPasswd.

- 4.1. Connectez-vous au cluster en tant qu'utilisateur `admin` pour vérifier que l'authentification HTPasswd est correctement configurée. L'opérateur d'authentification prend un certain temps pour charger les modifications de configuration de l'étape précédente.



#### Note

Si l'authentification échoue, patientez quelques instants et réessayez.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.
```

*...output omitted...*

- 4.2. Utilisez la commande `oc get nodes` pour vérifier que l'utilisateur `admin` dispose du rôle `cluster-admin`.

```
[student@workstation ~]$ oc get nodes
NAME      STATUS    ROLES          AGE     VERSION
master01   Ready     master,worker  2d     v1.23.5+9ce5071
master02   Ready     master,worker  2d     v1.23.5+9ce5071
master03   Ready     master,worker  2d     v1.23.5+9ce5071
```

- 4.3. Connectez-vous au cluster en tant qu'utilisateur `developer` pour vérifier que l'authentification HTPasswd est correctement configurée.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.
```

*...output omitted...*

**chapitre 10 |** Configuration de l'authentification et de l'autorisation

- 4.4. Utilisez la commande `oc get nodes` pour vérifier que les utilisateurs `developer` et `admin` ne partagent pas le même niveau d'accès.

```
[student@workstation ~]$ oc get nodes
Error from server (Forbidden): nodes is forbidden:
User "developer" cannot list resource "nodes" in API group "" at the cluster scope
```

- 4.5. Connectez-vous en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 4.6. Listez les utilisateurs actuels

```
[student@workstation ~]$ oc get users
NAME          UID                           FULL NAME  IDENTITIES
admin         31f6cccd2-6c58-47ee-978d-5e5e3c30d617
developer     d4e77b0d-9740-4f05-9af5-ecfc08a85101
```

- 4.7. Affichez la liste des identités actuelles.

```
[student@workstation ~]$ oc get identity
NAME           IDP NAME    IDP USER NAME   USER NAME   USER UID
myusers:admin  myusers     admin           admin       31f6cccd2-6c58-47...
myusers:developer  myusers     developer      developer   d4e77b0d-9740-4f...
```

- 5. En tant qu'utilisateur `admin`, créez un utilisateur HTPasswd nommé `manager` avec le mot de passe `redhat`.

- 5.1. Extrayez les données de fichier du secret vers le fichier `~/D0280/labs/auth-provider/htpasswd`.

```
[student@workstation ~]$ oc extract secret/localusers -n openshift-config \
>   --to ~/D0280/labs/auth-provider/ --confirm
/home/student/D0280/labs/auth-provider/htpasswd
```

- 5.2. Ajoutez une entrée dans votre fichier `~/D0280/labs/auth-provider/htpasswd` pour l'utilisateur supplémentaire `manager` avec le mot de passe `redhat`.

```
[student@workstation ~]$ htpasswd -b ~/D0280/labs/auth-provider/htpasswd \
>   manager redhat
Adding password for user manager
```

- 5.3. Examinez le contenu de votre fichier `~/D0280/labs/auth-provider/htpasswd` et vérifiez qu'il contient trois entrées avec des mots de passe hachés : une pour l'utilisateur `admin`, une pour `developer` et une pour `manager`.

```
[student@workstation ~]$ cat ~/DO280/labs/auth-provider/htpasswd
admin:$2y$05$QPuzHdl06IDkJssT.tdkZuSmgjUHV1XeYU4FjxhQrFqKL7hs2ZUL6
developer:$apr1$0Nzmc1rh$yGtne1k.JX6L5s5wNa2ye.
manager:$apr1$CJ/tpa6a$sLhjPkIIAy755ZArTT5EH/
```

- 5.4. Vous devez mettre à jour le secret après l'ajout d'utilisateurs supplémentaires. Utilisez la commande `oc set data secret` pour mettre à jour le secret. Si un échec se produit, exécutez à nouveau la commande après quelques instants, car il se peut que l'opérateur oauth soit toujours en cours de recharge.

```
[student@workstation ~]$ oc set data secret/localusers \
>   --from-file htpasswd=~/DO280/labs/auth-provider/htpasswd \
>   -n openshift-config
secret/localusers data updated
```

- 5.5. Attendez quelques instants que l'opérateur d'authentification se recharge, puis connectez-vous au cluster en tant qu'utilisateur `manager`.

**Note**

Si l'authentification échoue, patientez quelques instants et réessayez.

```
[student@workstation ~]$ oc login -u manager -p redhat
Login successful.

...output omitted...
```

- 6. Créez un projet nommé `auth-provider`, puis vérifiez que l'utilisateur `developer` ne peut pas accéder au projet.

- 6.1. En tant qu'utilisateur `manager`, créez un nouveau projet `auth-provider`.

```
[student@workstation ~]$ oc new-project auth-provider
Now using project "auth-provider" on server https://api.ocp4.example.com:6443".
...output omitted...
```

- 6.2. Connectez-vous en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.

...output omitted...
```

- 6.3. Tentative de suppression du projet `auth-provider`.

```
[student@workstation ~]$ oc delete project auth-provider
Error from server (Forbidden): projects.project.openshift.io "auth-provider"
is forbidden: User "developer" cannot delete resource "projects"
in API group "project.openshift.io" in the namespace "auth-provider"
```

- 7. Modifiez le mot de passe de l'utilisateur manager.

- 7.1. Connectez-vous en tant qu'utilisateur admin.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 7.2. Extrayez les données de fichier du secret vers le fichier ~/D0280/labs/auth-provider/htpasswd.

```
[student@workstation ~]$ oc extract secret/localusers -n openshift-config \
>   --to ~/D0280/labs/auth-provider/ --confirm
/home/student/D0280/labs/auth-provider/htpasswd
```

- 7.3. Générez un mot de passe utilisateur aléatoire et affectez-le à la variable MANAGER\_PASSWD.

```
[student@workstation ~]$ MANAGER_PASSWD="$(openssl rand -hex 15)"
```

- 7.4. Mettez à jour l'utilisateur manager pour qu'il utilise le mot de passe stocké dans la variable MANAGER\_PASSWD.

```
[student@workstation ~]$ htpasswd -b ~/D0280/labs/auth-provider/htpasswd \
>   manager ${MANAGER_PASSWD}
Updating password for user manager
```

- 7.5. Mettez à jour le secret.

```
[student@workstation ~]$ oc set data secret/localusers \
>   --from-file htpasswd=~/D0280/labs/auth-provider/htpasswd \
>   -n openshift-config
secret/localusers data updated
```

- 7.6. Connectez-vous en tant qu'utilisateur manager pour vérifier le mot de passe mis à jour.

```
[student@workstation ~]$ oc login -u manager -p ${MANAGER_PASSWD}
Login successful.

...output omitted...
```

**Note**

Si l'authentification échoue, patientez quelques instants et réessayez.

► 8. Supprimez l'utilisateur manager.

8.1. Connectez-vous en tant qu'utilisateur admin.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

8.2. Extrayez les données de fichier du secret vers le fichier ~/D0280/labs/auth-provider/htpasswd.

```
[student@workstation ~]$ oc extract secret/localusers -n openshift-config \
>   --to ~/D0280/labs/auth-provider/ --confirm
/home/student/D0280/labs/auth-provider/htpasswd
```

8.3. Supprimez l'utilisateur manager du fichier ~/D0280/labs/auth-provider/htpasswd.

```
[student@workstation ~]$ htpasswd -D ~/D0280/labs/auth-provider/htpasswd manager
Deleting password for user manager
```

8.4. Mettez à jour le secret.

```
[student@workstation ~]$ oc set data secret/localusers \
>   --from-file htpasswd=~/D0280/labs/auth-provider/htpasswd \
>   -n openshift-config
secret/localusers data updated
```

8.5. Supprimez la ressource Identity de l'utilisateur manager.

```
[student@workstation ~]$ oc delete identity "myusers:manager"
identity.user.openshift.io "myusers:manager" deleted
```

8.6. Supprimez la ressource User de l'utilisateur manager.

```
[student@workstation ~]$ oc delete user manager
user.user.openshift.io manager deleted
```

8.7. Dorénavant, toute tentative de connexion en tant qu'utilisateur manager échouera.

```
[student@workstation ~]$ oc login -u manager -p ${MANAGER_PASSWD}
Login failed (401 Unauthorized)
Verify you have provided correct credentials.
```

- 8.8. Listez les utilisateurs actuels pour vérifier que l'utilisateur `manager` est supprimé.

```
[student@workstation ~]$ oc get users
NAME          UID           FULL NAME  IDENTITIES
admin         31f6cccd2-6c58-47ee-978d-5e5e3c30d617   myusers:admin
developer    d4e77b0d-9740-4f05-9af5-ecfc08a85101   myusers:developer
```

- 8.9. Affichez la liste des identités actuelles pour vérifier que l'identité `manager` est supprimée.

```
[student@workstation ~]$ oc get identity
NAME          IDP NAME  IDP USER NAME  USER NAME
myusers:admin  myusers   admin          admin      ...
myusers:developer  myusers   developer     developer ...
```

- 8.10. Extrayez le secret et vérifiez que seuls les utilisateurs `admin` et `developer` s'affichent. La commande `--to -` envoie le secret à STDOUT au lieu de l'enregistrer dans un fichier.

```
[student@workstation ~]$ oc extract secret/localusers -n openshift-config --to -
# htpasswd
admin:$2y$05$TizWp/2ct4Edn08gmeMBI09IXujpLqkKAJ0Nldxc/V2XYYMBf6WBy
developer:$apr1$8Bc6txgb$bwHke4cGRGk9C8tQLg.hi1
```

## ► 9. Supprimez le fournisseur d'identité et nettoyez tous les utilisateurs.

- 9.1. Connectez-vous en tant qu'utilisateur `kubeadmin`.

```
[student@workstation ~]$ oc login -u kubeadmin -p ${RHT_OCP4_KUBEADM_PASSWD}
Login successful.
```

*...output omitted...*

- 9.2. Supprimez le projet `auth-provider`.

```
[student@workstation ~]$ oc delete project auth-provider
project.project.openshift.io "auth-provider" deleted
```

- 9.3. Modifiez la ressource en place pour supprimer le fournisseur d'identité d'OAuth :

```
[student@workstation ~]$ oc edit oauth
```

Supprimez toutes les lignes sous `spec:`, puis ajoutez `{}` après `spec:`. Ne modifiez aucune autre information dans le fichier. Votre ligne `spec:` doit correspondre à ce qui suit :

*...output omitted...*  
`spec: {}`

Enregistrez vos modifications, puis vérifiez que la commande `oc edit` les a appliquées :

```
oauth.config.openshift.io/cluster edited
```

9.4. Supprimez le secret localusers de l'espace de noms openshift-config.

```
[student@workstation ~]$ oc delete secret localusers -n openshift-config  
secret "localusers" deleted
```

9.5. Supprimez toutes les ressources user.

```
[student@workstation ~]$ oc delete user --all  
user.user.openshift.io "admin" deleted  
user.user.openshift.io "developer" deleted
```

9.6. Supprimez toutes les ressources identity.

```
[student@workstation ~]$ oc delete identity --all  
identity.user.openshift.io "myusers:admin" deleted  
identity.user.openshift.io "myusers:developer" deleted
```

## Fin

Sur la machine **workstation**, utilisez la commande **lab** pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab auth-provider finish
```

L'exercice guidé est maintenant terminé.

# Définition et application des permissions à l'aide des RBAC

---

## Résultats

À la fin de cette section, vous devez pouvoir définir les contrôles d'accès basés sur les rôles et d'appliquer des autorisations aux utilisateurs.

### Contrôle d'accès basé sur les rôles (RBAC, Role-Based Access Control),

Le contrôle d'accès basé sur les rôles (RBAC) est une technique qui permet de gérer l'accès aux ressources d'un système informatique. Dans Red Hat OpenShift, le RBAC détermine si un utilisateur peut effectuer certaines actions au sein du cluster ou du projet. Deux types de rôles peuvent être utilisés en fonction du niveau de responsabilité de l'utilisateur : cluster et local.



#### Note

L'autorisation est une étape distincte de l'authentification.

### Processus d'autorisation

Le processus d'autorisation est géré par des règles, des rôles et des liaisons.

Objet du RBAC	Description
Règle	Actions autorisées pour les objets ou groupes d'objets.
Rôle	Ensembles de règles. Les utilisateurs et les groupes peuvent être associés à plusieurs rôles.
Liaison	Affectation d'utilisateurs ou de groupes à un rôle.

### Étendue du RBAC

Red Hat OpenShift Container Platform (RHOC) définit deux groupes de rôles et de liaisons en fonction de l'étendue et de la responsabilité des utilisateurs : les rôles de cluster et les rôles locaux.

Niveau de rôle	Description
Cluster Role	Les utilisateurs ou groupes d'utilisateurs ayant ce niveau de rôle peuvent gérer le cluster OpenShift.
Local Role	Les utilisateurs ou les groupes d'utilisateurs ayant ce niveau de rôle ne peuvent gérer des éléments qu'au niveau d'un projet.

**Note**

Les liaisons des rôles de cluster sont prioritaires par rapport aux liaisons des rôles locaux.

## Gestion du RBAC à l'aide de l'interface de ligne de commande

Les administrateurs de cluster peuvent utiliser la commande `oc adm policy` pour ajouter et supprimer des rôles de cluster et des rôles d'espace de noms.

Pour ajouter un rôle de cluster à un utilisateur, utilisez la sous-commande `add-cluster-role-to-user` :

```
[user@host ~]$ oc adm policy add-cluster-role-to-user cluster-role username
```

Par exemple, pour changer un utilisateur standard en administrateur de cluster, utilisez la commande suivante :

```
[user@host ~]$ oc adm policy add-cluster-role-to-user cluster-admin username
```

Pour supprimer un rôle de cluster d'un utilisateur, utilisez la sous-commande `remove-cluster-role-from-user` :

```
[user@host ~]$ oc adm policy remove-cluster-role-from-user cluster-role username
```

Par exemple, pour changer un administrateur de cluster en utilisateur standard, utilisez la commande suivante :

```
[user@host ~]$ oc adm policy remove-cluster-role-from-user cluster-admin username
```

Les règles sont définies par une action et une ressource. Par exemple, la règle `create user` fait partie du rôle `cluster-admin`.

Vous pouvez utiliser la commande `oc adm policy who-can` pour déterminer si un utilisateur peut exécuter une action sur une ressource. Par exemple :

```
[user@host ~]$ oc adm policy who-can delete user
```

## Rôles par défaut

OpenShift est fourni avec un ensemble de rôles de cluster par défaut qui peuvent être affectés localement ou à l'ensemble du cluster. Vous pouvez modifier ces rôles pour un contrôle d'accès précis aux ressources OpenShift, mais il faut suivre des étapes supplémentaires qui ne sont pas décrites dans ce cours.

Rôles par défaut	Description
admin	Les utilisateurs ayant ce rôle peuvent gérer toutes les ressources d'un projet, notamment accorder l'accès au projet à d'autres utilisateurs.
basic-user	Les utilisateurs ayant ce rôle ont un accès en lecture au projet.
cluster-admin	Les utilisateurs ayant ce rôle ont un accès en tant que super-utilisateur aux ressources du cluster. Ces utilisateurs peuvent effectuer n'importe quelle action sur le cluster et ont un contrôle total sur tous les projets.
cluster-status	Les utilisateurs ayant ce rôle peuvent obtenir des informations sur l'état du cluster.
edit	Les utilisateurs ayant ce rôle peuvent créer, modifier et supprimer les ressources d'application courantes du projet, telles que des services et des déploiements. Ils ne peuvent pas agir sur les ressources de gestion telles que les plages de limite et les quotas et ne peuvent pas gérer les autorisations d'accès au projet.
self-provisioner	Les utilisateurs ayant ce rôle peuvent créer des projets. Il s'agit d'un rôle cluster et non d'un rôle projet.
view	Les utilisateurs ayant ce rôle peuvent afficher les ressources du projet, mais ils ne peuvent pas les modifier.

Le rôle `admin` octroie l'accès utilisateur aux ressources du projet, telles que les quotas et les plages de limite, ainsi que la capacité à créer des applications. Le rôle `edit` accorde aux utilisateurs un accès suffisant pour agir en tant que développeur à l'intérieur d'un projet, mais selon les contraintes configurées par un administrateur de projet.

Les administrateurs de cluster peuvent utiliser la commande `oc policy add-role-to-user` pour ajouter et supprimer des rôles d'espace de noms.

Ajoutez un rôle spécifié à un utilisateur à l'aide de la sous-commande `add-role-to-user`. Par exemple :

```
[user@host ~]$ oc policy add-role-to-user role-name username -n project
```

Par exemple, pour ajouter l'utilisateur `dev` au rôle `basic-user` dans le projet `wordpress` :

```
[user@host ~]$ oc policy add-role-to-user basic-user dev -n wordpress
```

## Types d'utilisateurs

L'interaction avec OpenShift Container Platform est associée à un utilisateur. Un objet `user` OpenShift Container Platform représente un utilisateur auquel vous pouvez accorder des autorisations dans le système en ajoutant des rôles (`roles`) via `rolebindings`. Il peut s'agir également d'un groupe d'utilisateurs.

## Utilisateurs standard

La plupart des utilisateurs interactifs d'OpenShift Container Platform sont des utilisateurs standard, représentés par l'objet `User`. Ce type d'utilisateur représente une personne autorisée à accéder à la plateforme.

Les exemples d'utilisateurs standard comprennent `user1` et `user2`.

## Utilisateurs système

Bon nombre des utilisateurs système sont automatiquement créés lorsque l'infrastructure est définie, principalement dans le but d'activer l'infrastructure afin qu'elle interagisse en toute sécurité avec l'API. Les utilisateurs système comprennent un administrateur de cluster (qui peut accéder à tout), un utilisateur par nœud, des utilisateurs pour les routeurs et les registres, et bien plus encore. Un utilisateur système anonyme est utilisé par défaut pour les demandes non authentifiées.

Les exemples d'utilisateurs système sont notamment `system:admin`, `system:openshift-registry` et `system:node:node1.example.com`.

## Comptes de service

Il s'agit d'utilisateurs système spéciaux associés à des projets. Certains utilisateurs de compte de service sont créés automatiquement lors de la création du projet. Les administrateurs de projet peuvent en créer d'autres dans le but de définir l'accès au contenu de chaque projet. Les comptes de service sont souvent utilisés pour accorder des privilèges supplémentaires aux pods ou aux déploiements. Les comptes de service sont représentés par l'objet `ServiceAccount`.

Les exemples d'utilisateurs comptes de service sont `system:serviceaccount:default:deployer` et `system:serviceaccount:foo:builder`.

Chaque utilisateur doit s'authentifier avant de pouvoir accéder à OpenShift Container Platform. Les demandes d'API sans authentification ou avec une authentification non valide sont authentifiées en tant que demandes émanant de l'utilisateur système anonyme. Une fois l'authentification réussie, la politique détermine ce que l'utilisateur est autorisé à faire.



## Références

Pour plus d'informations sur les espaces de noms Kubernetes, reportez-vous à **Documentation de Kubernetes**

<https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>

Pour plus d'informations sur RBAC, reportez-vous au chapitre *Using RBAC to define and apply permissions* de la documentation *Authentication and authorization* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/authentication\\_and\\_authorization/index#using-rbac](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/authentication_and_authorization/index#using-rbac)

Pour plus d'informations sur les groupes, reportez-vous au chapitre *Understanding authentication* de la documentation *Authentication and authorization* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/authentication\\_and\\_authorization/index#understanding-authentication](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/authentication_and_authorization/index#understanding-authentication)

## ► Exercice guidé

# Définition et application des permissions à l'aide des RBAC

Dans cet exercice, vous allez définir les contrôles d'accès basés sur les rôles et appliquer des autorisations aux utilisateurs.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Supprimer les privilèges de création de projet des utilisateurs qui ne sont pas des administrateurs de cluster OpenShift.
- Créer des groupes OpenShift et ajouter des membres à ces groupes.
- Créer un projet et lui attribuer des privilèges d'administration de projet.
- En tant qu'administrateur de projet, attribuez des privilèges de lecture et d'écriture à différents groupes d'utilisateurs.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée certains utilisateurs HTPasswd nécessaires à cet exercice.

```
[student@workstation ~]$ lab auth-rbac start
```

## Instructions

- 1. Connectez-vous au cluster OpenShift et déterminez les liaisons de rôle de cluster qui affectent le rôle de cluster `self-provisioner`.

- 1.1. Connectez-vous au cluster en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
>     https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Répertoriez toutes les liaisons de rôle de cluster qui font référence au rôle de cluster `self-provisioner`.

```
[student@workstation ~]$ oc get clusterrolebinding -o wide | \
>   grep -E 'NAME|self-provisioner'
NAME                  ROLE
self-provisioners    ...   ClusterRole/self-provisioner  ...
...
```

- ▶ 2. Retirez le privilège permettant de créer des projets à tous les utilisateurs qui ne sont pas des administrateurs de cluster en supprimant le rôle de cluster **self-provisioner** du groupe virtuel **system:authenticated:oauth**.
- 2.1. Vérifiez que la liaison de rôle de cluster **self-provisioners** que vous avez trouvée à l'étape précédente affecte le rôle de cluster **self-provisioner** au groupe **system:authenticated:oauth**.

```
[student@workstation ~]$ oc describe clusterrolebindings self-provisioners
Name:          self-provisioners
Labels:        <none>
Annotations:  rbac.authorization.kubernetes.io/autoupdate: true
Role:
  Kind:  ClusterRole
  Name:  self-provisioner
Subjects:
  Kind  Name           Namespace
  ----  ---           -----
  Group  system:authenticated:oauth
```

- 2.2. Supprimez le rôle de cluster **self-provisioner** du groupe virtuel **system:authenticated:oauth**, ce qui supprime la liaison de rôle **self-provisioners**.

```
[student@workstation ~]$ oc adm policy remove-cluster-role-from-group \
>   self-provisioner system:authenticated:oauth
Warning: Your changes may get lost whenever a master is restarted, unless you
prevent reconciliation of this rolebinding using the following command:
oc annotate clusterrolebinding.rbac self-provisioner
'rbac.authorization.kubernetes.io/autoupdate=false' --overwrite
clusterrole.rbac.authorization.k8s.io/self-provisioner removed:
"system:authenticated:oauth"
```



### Note

Vous pouvez ignorer en toute sécurité l'avertissement indiquant que vos modifications seront perdues.

- 2.3. Vérifiez que le rôle a été supprimé du groupe. Il ne doit pas y avoir de liaison de rôle de cluster **self-provisioners**.

```
[student@workstation ~]$ oc describe clusterrolebindings self-provisioners
Error from server (NotFound): clusterrolebindings.rbac.authorization.k8s.io "self-
provisioners" not found
```

- 2.4. Déterminez si d'autres liaisons de rôle de cluster font référence au rôle de cluster **self-provisioner**.

```
[student@workstation ~]$ oc get clusterrolebinding -o yaml | \
>   grep -E 'NAME|self-provisioner'
NAME                ROLE      ...

```

- 2.5. Connectez-vous en tant qu'utilisateur **leader** avec le mot de passe **redhat**.

```
[student@workstation ~]$ oc login -u leader -p redhat
Login successful.

...output omitted...
```

- 2.6. Essayez de créer un projet, l'opération doit échouer.

```
[student@workstation ~]$ oc new-project test
Error from server (Forbidden): You may not request a new project via this API.
```

► 3. Créez un projet et ajoutez des privilèges d'administration de projet à l'utilisateur **leader**.

- 3.1. Connectez-vous en tant qu'utilisateur **admin**.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 3.2. Créez le projet **auth-rbac**.

```
[student@workstation ~]$ oc new-project auth-rbac
Now using project "auth-rbac" on server "https://api.ocp4.example.com:6443".

...output omitted...
```

- 3.3. Accordez des privilèges d'administration de projet à l'utilisateur **leader** sur le projet **auth-rbac**.

```
[student@workstation ~]$ oc policy add-role-to-user admin leader
clusterrole.rbac.authorization.k8s.io/admin added: "leader"
```

► 4. Créez les groupes **dev-group** et **qa-group**, et ajoutez leurs membres respectifs.

- 4.1. Créez un groupe nommé **dev-group**.

```
[student@workstation ~]$ oc adm groups new dev-group
group.user.openshift.io/dev-group created
```

- 4.2. Ajoutez l'utilisateur **developer** à **dev-group**.

```
[student@workstation ~]$ oc adm groups add-users dev-group developer
group.user.openshift.io/dev-group added: "developer"
```

4.3. Créez un deuxième groupe nommé qa-group.

```
[student@workstation ~]$ oc adm groups new qa-group
group.user.openshift.io/qa-group created
```

4.4. Ajoutez l'utilisateur qa-engineer à qa-group.

```
[student@workstation ~]$ oc adm groups add-users qa-group qa-engineer
group.user.openshift.io/qa-group added: "qa-engineer"
```

4.5. Examinez tous les groupes OpenShift existants pour vérifier que les membres qui les composent sont les bons.

```
[student@workstation ~]$ oc get groups
NAME      USERS
dev-group  developer
qa-group   qa-engineer
```

- 5. En tant qu'utilisateur leader, attribuez les priviléges d'écriture au groupe dev-group et les priviléges de lecture au groupe qa-group sur le projet auth-rbac.

5.1. Connectez-vous en tant qu'utilisateur leader.

```
[student@workstation ~]$ oc login -u leader -p redhat
Login successful.

...output omitted...

Using project "auth-rbac".
```

5.2. Ajoutez des priviléges d'écriture à dev-group sur le projet auth-rbac.

```
[student@workstation ~]$ oc policy add-role-to-group edit dev-group
clusterrole.rbac.authorization.k8s.io/edit added: "dev-group"
```

5.3. Ajoutez des priviléges de lecture à qa-group sur le projet auth-rbac.

```
[student@workstation ~]$ oc policy add-role-to-group view qa-group
clusterrole.rbac.authorization.k8s.io/view added: "qa-group"
```

5.4. Examinez toutes les liaisons de rôle sur le projet auth-rbac pour vérifier qu'elles affectent des rôles aux groupes et utilisateurs appropriés. La sortie ci-dessous omet les liaisons de rôle par défaut assignées par OpenShift aux comptes de service.

```
[student@workstation ~]$ oc get rolebindings -o wide
NAME      ROLE          AGE   USERS   GROUPS
admin     ClusterRole/admin  58s   admin
admin-0   ClusterRole/admin  51s   leader
edit      ClusterRole/edit   12s
...output omitted...
view      ClusterRole/view   8s
                                qa-group
```

- ▶ 6. En tant qu'utilisateur `developer`, déployez un serveur HTTP Apache pour confirmer que l'utilisateur `developer` dispose de privilèges d'écriture dans le projet. Essayez également d'accorder des privilèges d'écriture à l'utilisateur `qa-engineer` pour confirmer que l'utilisateur `developer` n'a pas de privilèges d'administration de projet.

- 6.1. Connectez-vous en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.

...output omitted...

Using project "auth-rbac".
```

- 6.2. Déployez un serveur HTTP Apache à l'aide du flux d'images standard d'OpenShift.

```
[student@workstation ~]$ oc new-app --name httpd httpd:2.4
...output omitted...
--> Creating resources ...
imagestreamtag.image.openshift.io "httpd:2.4" created
deployment.apps "httpd" created
service "httpd" created
--> Success
...output omitted...
```

- 6.3. Essayez d'accorder des privilèges d'écriture à l'utilisateur `qa-engineer`, l'opération doit échouer.

```
[student@workstation ~]$ oc policy add-role-to-user edit qa-engineer
Error from server (Forbidden): rolebindings.rbac.authorization.k8s.io is
forbidden: User "developer" cannot list resource "rolebindings" in API group
"rbac.authorization.k8s.io" in the namespace "auth-rbac"
```

- ▶ 7. Vérifiez que l'utilisateur `qa-engineer` ne dispose que de privilèges de lecture sur l'application `httpd`.

- 7.1. Connectez-vous en tant qu'utilisateur `qa-engineer`.

```
[student@workstation ~]$ oc login -u qa-engineer -p redhat
Login successful.

...output omitted...

Using project "auth-rbac".
```

7.2. Essayez de mettre à l'échelle l'application httpd, l'opération doit échouer.

```
[student@workstation ~]$ oc scale deployment httpd --replicas 3
Error from server (Forbidden): deployments.apps "httpd" is forbidden: User "qa-
engineer" cannot patch resource "deployments/scale" in API group "apps" in the
namespace "auth-rbac"
```

► **8.** Restaurez les privilèges de création de projet pour tous les utilisateurs.

8.1. Connectez-vous en tant qu'utilisateur admin.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

8.2. Restaurez les privilèges de création de projet pour tous les utilisateurs en recréant la liaison de rôle de cluster self-provisioners créée par le programme d'installation d'OpenShift.

```
[student@workstation ~]$ oc adm policy add-cluster-role-to-group \
>   --rolebinding-name self-provisioners \
>   self-provisioner system:authenticated:oauth
Warning: Group 'system:authenticated:oauth' not found
clusterrole.rbac.authorization.k8s.io/self-provisioner added:
"system:authenticated:oauth"
```

**Note**

Vous pouvez ignorer en toute sécurité l'avertissement indiquant que le groupe est introuvable.

## Fin

Sur la machine workstation, exécutez la commande lab pour mettre fin à l'exercice.

```
[student@workstation ~]$ lab auth-rbac finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Configuration de l'authentification et de l'autorisation

### Liste de contrôle des performances

Au cours de cet atelier, vous allez configurer le fournisseur d'identité HTPasswd, créer des groupes, et attribuer des rôles à des utilisateurs et des groupes.

### Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Créer des utilisateurs et des mots de passe pour l'authentification HTPasswd.
- Configurer le fournisseur d'identité pour l'authentification HTPasswd.
- Affecter des droits d'administration de cluster aux utilisateurs.
- Supprimer la capacité de créer des projets au niveau du cluster.
- Créer des groupes et ajouter des utilisateurs à ces groupes.
- Gérer les priviléges des utilisateurs dans les projets en accordant des priviléges aux groupes.

### Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

```
[student@workstation ~]$ lab auth-review start
```

Cette commande garantit que l'API du cluster est accessible, que le paquetage `httpd-util` est installé et que les paramètres d'authentification sont configurés selon les valeurs d'installation par défaut.

### Instructions

1. Mettez à jour le fichier d'authentification HTPasswd `~/D0280/labs/auth-review/tmp_users` existant pour supprimer l'utilisateur `analyst`. Assurez-vous que les utilisateurs `tester` et `leader` dans le fichier utilisent le mot de passe `L@bR3v!ew`. Ajoutez deux nouvelles entrées au fichier pour les utilisateurs `admin` et `developer`. Utilisez `L@bR3v!ew` comme mot de passe pour chaque nouvel utilisateur.
2. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `kubeadmin` à l'aide de la variable `RHT_OCP4_KUBEADM_PASSWD` définie dans le fichier `/usr/local/etc/ocp4.config` en tant que mot de passe. Configurez votre cluster pour utiliser le fournisseur d'identité HTPasswd à l'aide des noms d'utilisateur et des mots de passe définis dans le fichier `~/D0280/labs/auth-review/tmp_users`.

3. Faites de l'utilisateur `admin` un administrateur de cluster. Connectez-vous en tant qu'utilisateur `admin` et `developer` pour vérifier la configuration utilisateur HTPasswd et les priviléges de cluster.
4. En tant qu'utilisateur `admin`, supprimez la capacité de créer des projets à l'échelle du cluster.
5. Créez un groupe nommé `managers` et ajoutez-y l'utilisateur `leader`. Accordez des priviléges de création de projet au groupe `managers`. En tant qu'utilisateur `leader`, créez le projet `auth-review`.
6. Créez un groupe nommé `developers` et accordez des priviléges de modification sur le projet `auth-review`. Ajoutez l'utilisateur `developer` au groupe.
7. Créez un groupe nommé `qa` et accordez des priviléges d'affichage sur le projet `auth-review`. Ajoutez l'utilisateur `tester` au groupe.

## Évaluation

Sur la machine `workstation`, exécutez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab auth-review grade
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab auth-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Configuration de l'authentification et de l'autorisation

### Liste de contrôle des performances

Au cours de cet atelier, vous allez configurer le fournisseur d'identité HTPasswd, créer des groupes, et attribuer des rôles à des utilisateurs et des groupes.

### Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Créer des utilisateurs et des mots de passe pour l'authentification HTPasswd.
- Configurer le fournisseur d'identité pour l'authentification HTPasswd.
- Affecter des droits d'administration de cluster aux utilisateurs.
- Supprimer la capacité de créer des projets au niveau du cluster.
- Créer des groupes et ajouter des utilisateurs à ces groupes.
- Gérer les priviléges des utilisateurs dans les projets en accordant des priviléges aux groupes.

### Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

```
[student@workstation ~]$ lab auth-review start
```

Cette commande garantit que l'API du cluster est accessible, que le paquetage `httpd-util` est installé et que les paramètres d'authentification sont configurés selon les valeurs d'installation par défaut.

### Instructions

1. Mettez à jour le fichier d'authentification HTPasswd `~/D0280/labs/auth-review/tmp_users` existant pour supprimer l'utilisateur `analyst`. Assurez-vous que les utilisateurs `tester` et `leader` dans le fichier utilisent le mot de passe `L@bR3v!ew`. Ajoutez deux nouvelles entrées au fichier pour les utilisateurs `admin` et `developer`. Utilisez `L@bR3v!ew` comme mot de passe pour chaque nouvel utilisateur.
  - 1.1. Supprimez l'utilisateur `analyst` du fichier d'authentification HTPasswd `~/D0280/labs/auth-review/tmp_users`.

```
[student@workstation ~]$ htpasswd -D ~/D0280/labs/auth-review/tmp_users analyst
Deleting password for user analyst
```

- 1.2. Mettez à jour les entrées relatives aux utilisateurs `tester` et `leader` de sorte qu'ils utilisent le mot de passe `L@bR3v!ew`. Ajoutez des entrées pour les utilisateurs `admin` et `developer` à l'aide du mot de passe `L@bR3v!ew`.

```
[student@workstation ~]$ for NAME in tester leader admin developer
>   do
>     htpasswd -b ~/D0280/labs/auth-review/tmp_users ${NAME} 'L@bR3v!ew'
>   done
Updating password for user tester
Updating password for user leader
Adding password for user admin
Adding password for user developer
```

- 1.3. Parcourez le contenu du fichier `~/D0280/labs/auth-review/tmp_users`. Il ne contient pas de ligne pour l'utilisateur `analyst`. Il inclut deux nouvelles entrées avec des mots de passe hachés pour les utilisateurs `admin` et `developer`.

```
[student@workstation ~]$ cat ~/D0280/labs/auth-review/tmp_users
tester:$apr1$0eqhKgbU$DwD0CB4IumhasaRuEr6hp0
leader:$apr1$.EB5IXlu$FDV.Av16nj10CMzgolScr
admin:$apr1$ItcCncDS$xFQCUjQGTsXAup00KQfmw0
developer:$apr1$D8F1Hren$izDhAWq5DRjUHPv0i7FHn.
```

2. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `kubeadmin` à l'aide de la variable `RHT_OCP4_KUBEADM_PASSWD` définie dans le fichier `/usr/local/etc/ocp4.config` en tant que mot de passe. Configurez votre cluster pour utiliser le fournisseur d'identité `HTPasswd` à l'aide des noms d'utilisateur et des mots de passe définis dans le fichier `~/D0280/labs/auth-review/tmp_users`.
- 2.1. Procurez-vous le fichier de configuration de la salle de classe, accessible à l'adresse `/usr/local/etc/ocp4.config`.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 2.2. Connectez-vous au cluster en tant qu'utilisateur `kubeadmin`.

```
[student@workstation ~]$ oc login -u kubeadmin -p ${RHT_OCP4_KUBEADM_PASSWD} \
>   https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 2.3. Créez un secret nommé `auth-review` à l'aide du fichier `~/D0280/labs/auth-review/tmp_users`.

```
[student@workstation ~]$ oc create secret generic auth-review \
>   --from-file htpasswd=/home/student/D0280/labs/auth-review/tmp_users \
>   -n openshift-config
secret/auth-review created
```

- 2.4. Exportez la ressource OAuth existante vers `~/D0280/labs/auth-review/oauth.yaml`.

```
[student@workstation ~]$ oc get oauth cluster \
> -o yaml > ~/D0280/labs/auth-review/oauth.yaml
```

- 2.5. Modifiez le fichier ~/D0280/labs/auth-review/oauth.yaml de manière à remplacer la ligne spec: {} par les lignes en gras suivantes. Notez que htpasswd, mappingMethod, name et type se trouvent au même niveau de mise en retrait.

```
apiVersion: config.openshift.io/v1
kind: OAuth
...output omitted...
spec:
  identityProviders:
    - htpasswd:
        fileData:
          name: auth-review
        mappingMethod: claim
        name: htpasswd
        type: HTPasswd
```

**Note**

Pour plus de commodité, le fichier ~/D0280/solutions/auth-review/oauth.yaml contient une version minimale de la configuration OAuth avec les personnalisations spécifiées.

- 2.6. Appliquez la ressource personnalisée définie à l'étape précédente.

```
[student@workstation ~]$ oc replace -f ~/D0280/labs/auth-review/oauth.yaml
oauth.config.openshift.io/cluster replaced
```

- 2.7. Une mise à jour réussie de la ressource oauth/cluster recrée les pods oauth-openshift dans l'espace de noms openshift-authentication.

```
[student@workstation ~]$ watch oc get pods -n openshift-authentication
```

Attendez que les nouveaux pods oauth-openshift soient prêts et en cours d'exécution, et que les pods précédents soient terminés.

NAME	READY	STATUS	RESTARTS	AGE
oauth-openshift-6755d8795-h8bgv	1/1	Running	0	34s
oauth-openshift-6755d8795-rk4m6	1/1	Running	0	38s
oauth-openshift-6755d8795-2859w	1/1	Running	0	53s

Appuyez sur Ctrl+C pour quitter la commande watch.

3. Faites de l'utilisateur **admin** un administrateur de cluster. Connectez-vous en tant qu'utilisateur **admin** et **developer** pour vérifier la configuration utilisateur HTPasswd et les priviléges de cluster.

- 3.1. Assignez le rôle **admin** à l'utilisateur **cluster-admin**.

```
[student@workstation ~]$ oc adm policy add-cluster-role-to-user \
>   cluster-admin admin
Warning: User 'admin' not found
clusterrole.rbac.authorization.k8s.io/cluster-admin added: "admin"
```

**Note**

Vous pouvez ignorer en toute sécurité l'avertissement indiquant que l'utilisateur `admin` est introuvable.

- 3.2. Connectez-vous au cluster en tant qu'utilisateur `admin` pour vérifier que l'authentification HTPasswd a été correctement configurée.

```
[student@workstation ~]$ oc login -u admin -p 'L@bR3v!ew'
Login successful.

...output omitted...
```

- 3.3. Utilisez la commande `oc get nodes` pour vérifier que l'utilisateur `admin` dispose du rôle `cluster-admin`. Les noms des nœuds de votre cluster peuvent être différents.

```
[student@workstation ~]$ oc get nodes
NAME      STATUS    ROLES          AGE     VERSION
master01   Ready     master,worker  4d      v1.23.5+9ce5071
master02   Ready     master,worker  4d      v1.23.5+9ce5071
master03   Ready     master,worker  4d      v1.23.5+9ce5071
```

- 3.4. Connectez-vous au cluster en tant qu'utilisateur `developer` pour vérifier que l'authentification HTPasswd est correctement configurée.

```
[student@workstation ~]$ oc login -u developer -p 'L@bR3v!ew'
Login successful.

...output omitted...
```

- 3.5. Utilisez la commande `oc get nodes` pour vérifier que l'utilisateur `developer` ne dispose pas des priviléges d'administration de cluster.

```
[student@workstation ~]$ oc get nodes
Error from server (Forbidden): nodes is forbidden: User "developer" cannot list
resource "nodes" in API group "" at the cluster scope
```

4. En tant qu'utilisateur `admin`, supprimez la capacité de créer des projets à l'échelle du cluster.

- 4.1. Connectez-vous au cluster en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p 'L@bR3v!ew'
Login successful.

...output omitted...
```

- 4.2. Supprimez le rôle de cluster **self-provisioner** du groupe virtuel **system:authenticated:oauth**.

```
[student@workstation ~]$ oc adm policy remove-cluster-role-from-group \
>   self-provisioner system:authenticated:oauth
clusterrole.rbac.authorization.k8s.io/self-provisioner removed:
"system:authenticated:oauth"
```



### Note

Vous pouvez ignorer en toute sécurité l'avertissement indiquant que vos modifications seront perdues.

5. Créez un groupe nommé **managers** et ajoutez-y l'utilisateur **leader**. Accordez des priviléges de création de projet au groupe **managers**. En tant qu'utilisateur **leader**, créez le projet **auth-review**.

- 5.1. Créez un groupe nommé **managers**.

```
[student@workstation ~]$ oc adm groups new managers
group.user.openshift.io/managers created
```

- 5.2. Ajoutez l'utilisateur **leader** au groupe **managers**.

```
[student@workstation ~]$ oc adm groups add-users managers leader
group.user.openshift.io/managers added: "leader"
```

- 5.3. Affectez le rôle de cluster **self-provisioner** au groupe **managers**.

```
[student@workstation ~]$ oc adm policy add-cluster-role-to-group \
>   self-provisioner managers
clusterrole.rbac.authorization.k8s.io/self-provisioner added: "managers"
```

- 5.4. En tant qu'utilisateur **leader**, créez le projet **auth-review**.

```
[student@workstation ~]$ oc login -u leader -p 'L@bR3v!ew'
Login successful.

...output omitted...
```

L'utilisateur qui crée un projet se voit attribuer automatiquement le rôle **admin** sur le projet en question.

```
[student@workstation ~]$ oc new-project auth-review
Now using project "auth-review" on server "https://api.ocp4.example.com:6443".

...output omitted...
```

6. Créez un groupe nommé **developers** et accordez des priviléges de modification sur le projet **auth-review**. Ajoutez l'utilisateur **developer** au groupe.

- 6.1. Connectez-vous au cluster en tant qu'utilisateur **admin**.

```
[student@workstation ~]$ oc login -u admin -p 'L@bR3v!ew'  
Login successful.  
...output omitted...
```

6.2. Créez un groupe nommé **developers**.

```
[student@workstation ~]$ oc adm groups new developers  
group.user.openshift.io/developers created
```

6.3. Ajoutez l'utilisateur **developer** au groupe **developers**.

```
[student@workstation ~]$ oc adm groups add-users developers developer  
group.user.openshift.io/developers added: "developer"
```

6.4. Accordez des priviléges de modification au groupe **developers** sur le projet **auth-review**.

```
[student@workstation ~]$ oc policy add-role-to-group edit developers  
clusterrole.rbac.authorization.k8s.io/edit added: "developers"
```

7. Créez un groupe nommé **qa** et accordez des priviléges d'affichage sur le projet **auth-review**. Ajoutez l'utilisateur **tester** au groupe.

7.1. Créez un groupe nommé **qa**.

```
[student@workstation ~]$ oc adm groups new qa  
group.user.openshift.io/qa created
```

7.2. Ajoutez l'utilisateur **tester** au groupe **qa**.

```
[student@workstation ~]$ oc adm groups add-users qa tester  
group.user.openshift.io/qa added: "tester"
```

7.3. Accordez des priviléges d'affichage au groupe **qa** sur le projet **auth-review**.

```
[student@workstation ~]$ oc policy add-role-to-group view qa  
clusterrole.rbac.authorization.k8s.io/view added: "qa"
```

## Évaluation

Sur la machine **workstation**, exécutez la commande **lab** pour évaluer votre travail. Corrigez toute erreur signalée et répétez le script tant que des erreurs persistent.

```
[student@workstation ~]$ lab auth-review grade
```

## Fin

Sur la machine `workstation`, utilisez la commande `Lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab auth-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Un cluster OpenShift nouvellement installé offre deux méthodes d'authentification qui accordent un accès administratif : le fichier `kubeconfig` et l'utilisateur virtuel `kubeadm`.
- Le fournisseur d'identité HTPasswd authentifie les utilisateurs en fonction des informations d'identification stockées dans un secret. Le nom du secret, ainsi que les autres paramètres du fournisseur d'identité, sont stockés dans la ressource personnalisée OAuth.
- Pour gérer les informations d'identification de l'utilisateur à l'aide du fournisseur d'identité HTPasswd, vous devez extraire les données du secret, modifier ces données à l'aide de la commande `htpasswd`, puis appliquer à nouveau les données au secret.
- La création d'utilisateurs OpenShift requiert des informations d'identification valides, gérées par un fournisseur d'identité, ainsi que des ressources User et Identity.
- La suppression des utilisateurs OpenShift nécessite de supprimer leurs informations d'identification du fournisseur d'identité, et de supprimer également leurs ressources User et Identity.
- OpenShift utilise le contrôle d'accès basé sur les rôles (RBAC) pour contrôler les actions de l'utilisateur. Un rôle est un ensemble de règles qui régissent l'interaction avec les ressources OpenShift. Il existe des rôles par défaut pour les administrateurs de cluster, les développeurs et les auditeurs.
- Pour contrôler l'interaction avec l'utilisateur, attribuez un ou plusieurs rôles à un utilisateur. Une liaison de rôle contient toutes les associations entre un rôle et des utilisateurs et des groupes.
- Pour accorder des privilèges d'administrateur de cluster à un utilisateur, affectez le rôle `cluster-admin` à cet utilisateur.

# Configuration de la sécurité des applications

## Objectif

Limiter les autorisations des applications à l'aide de contraintes de contexte de sécurité et protéger les informations d'accès à l'aide de secrets.

## Résultats

- Créer et appliquer des secrets pour gérer les informations sensibles, et partagez des secrets entre les applications.
- Créer des comptes de service, appliquer des autorisations et gérer les contraintes de contexte de sécurité.

## Sections

- Gestion des informations sensibles à l'aide des secrets (et exercice guidé)
- Contrôle des permissions d'application à l'aide des contraintes du contexte de sécurité (et exercice guidé)

## Atelier

Configuration de la sécurité des applications

# Gestion des informations sensibles à l'aide des secrets

---

## Résultats

À la fin de cette section, vous devez pouvoir créer et appliquer des secrets pour gérer des informations sensibles, et partager des secrets entre les applications.

## Présentation des secrets

Les applications modernes sont conçues pour associer librement le code, la configuration et les données. Les fichiers de configuration et les données ne sont pas codés en dur en tant que partie intégrante du logiciel. Le logiciel charge plutôt la configuration et les données à partir d'une source externe. Cela permet de déployer une application dans différents environnements sans avoir à modifier le code source de l'application.

Souvent, les applications ont besoin d'accéder à des informations sensibles. Par exemple, une application Web backend doit avoir accès aux informations d'identification de la base de données pour effectuer une requête de base de données. Kubernetes et OpenShift utilisent des ressources secrètes pour détenir des informations sensibles, telles que :

- les mots de passe ;
- les fichiers de configuration sensibles ;
- les informations d'identification à une ressource externe, telles qu'une clé SSH ou un jeton OAuth.

Un secret peut stocker n'importe quel type de données. Les données d'un secret sont codées en Base64 ; elles ne sont pas stockées en texte clair. Les données secrètes ne sont pas chiffrées ; vous pouvez décoder le secret au format Base64 pour accéder aux données d'origine.

Bien que les secrets puissent stocker n'importe quel type de données, Kubernetes et OpenShift prennent en charge différents types de secrets. Il existe différents types de ressources secrètes, notamment les jetons de compte de service, les clés SSH et les certificats TLS. Lorsque vous stockez des informations dans un type de ressource secrète spécifique, Kubernetes confirme que les données sont conformes au type de secret.



### Note

Vous pouvez chiffrer la base de données Etcd, même si l'option n'est pas activée par défaut. Lorsque cette option est activée, Etcd chiffre les ressources suivantes : secrets, mappages de configuration, routes, tokens d'accès OAuth et tokens d'autorisation OAuth. L'activation du chiffrement Etcd n'entre pas dans le cadre de cette formation.

## Caractéristiques des secrets

Les principales caractéristiques des secrets sont les suivantes :

- Les données secrètes peuvent être partagées au sein d'un espace de noms de projet.

- Les données secrètes sont référencées indépendamment de la définition du secret. Les administrateurs peuvent créer et gérer une ressource secrète à laquelle les autres membres de l'équipe peuvent faire référence dans leurs configurations de déploiement.
- Les données secrètes sont injectées dans des pods lorsqu'OpenShift crée un pod. Vous pouvez exposer un secret en tant que variable d'environnement ou en tant que fichier monté dans le pod.
- Si la valeur d'un secret change pendant l'exécution du pod, les données secrètes ne seront pas mises à jour dans le pod. Dès qu'une valeur de secret est modifiée, vous devez créer des pods pour injecter les nouvelles données secrètes.
- Toutes les données secrètes qu'OpenShift injecte dans un pod sont éphémères. Si OpenShift expose des données sensibles à un pod en tant que variables d'environnement, ces variables sont détruites à la destruction du pod.

Les volumes de données secrètes sont pris en charge par un stockage de fichiers temporaire. Si un secret est monté sous la forme d'un fichier dans le pod, le fichier en question est également détruit à la destruction du pod. Un pod arrêté ne contient pas de données secrètes.

## Cas d'utilisation des secrets

Les deux principaux cas d'utilisation des secrets sont le stockage des informations d'identification et la sécurisation des communications entre les services.

### Credentials

Stockez des informations sensibles, telles que des mots de passe et des noms d'utilisateur, dans un secret.

Si une application s'attend à lire des informations sensibles à partir d'un fichier, vous montez le secret en tant que volume de données dans le pod. L'application peut lire le secret en tant que fichier ordinaire pour accéder aux informations sensibles. Certaines bases de données, par exemple, lisent les informations d'identification d'un fichier pour authentifier les utilisateurs.

Certaines applications utilisent des variables d'environnement pour lire la configuration et les données sensibles. Vous pouvez lier des variables de secret à des variables d'environnement du pod dans une configuration de déploiement.

### TLS (Transport Layer Security) et paires de clés

Utilisez un certificat TLS et une clé pour sécuriser la communication avec un pod. Un secret TLS stocke le certificat sous la forme `tls.crt` et la clé de certificat en tant que `tls.key`. Les développeurs peuvent monter le secret en tant que volume et créer une route directe (pass-through) vers l'application.

### Création d'un secret

Si un pod a besoin d'accéder à des informations sensibles, créez un secret pour ces informations avant de déployer le pod. Utilisez l'une des commandes suivantes pour créer un secret :

- Créez un secret générique contenant des paires clé-valeur à partir des valeurs littérales saisies sur la ligne de commande :

```
[user@host ~]$ oc create secret generic secret_name \
>   --from-literal key1=secret1 \
>   --from-literal key2=secret2
```

- Créez un secret générique à l'aide des noms de clé spécifiés sur la ligne de commande et des valeurs à partir des fichiers :

```
[user@host ~]$ oc create secret generic ssh-keys \
>   --from-file id_rsa=/path-to/id_rsa \
>   --from-file id_rsa.pub=/path-to/id_rsa.pub
```

- Créez un secret TLS spécifiant un certificat et la clé associée :

```
[user@host ~]$ oc create secret tls secret-tls \
>   --cert /path-to-certificate --key /path-to-key
```

## Exposition des secrets aux pods

Pour exposer un secret à un pod, commencez par créer le secret. Affectez chaque élément des données sensibles à une clé. Après la création, le secret contient des paires clé-valeur. La commande suivante crée un secret générique nommé **demo-secret** avec deux clés : **user** avec la valeur **demo-user** et **root\_password** avec la valeur **zT1KTgk**.

```
[user@host ~]$ oc create secret generic demo-secret \
>   --from-literal user=demo-user
>   --from-literal root_password=zT1KTgk
```

## Secrets sous forme de variables d'environnement du pod

Prenons l'exemple d'une application de base de données qui lit le mot de passe de l'administrateur de base de données dans la variable d'environnement **MYSQL\_ROOT\_PASSWORD**. Modifiez la section des variables d'environnement de la configuration de déploiement pour utiliser les valeurs du secret :

```
env:
  - name: MYSQL_ROOT_PASSWORD ①
    valueFrom:
      secretKeyRef: ②
        name: demo-secret ③
        key: root_password ④
```

- ❶ Nom de la variable d'environnement dans le pod, qui contient les données d'un secret.
- ❷ Clé **secretKeyRef** qui attend un secret. Utilisez la clé **configMapKeyRef** pour les mappages de configuration.
- ❸ Nom du secret qui contient les informations sensibles souhaitées.
- ❹ Nom de la clé qui contient les informations sensibles dans le secret.

Vous pouvez également utiliser la commande `oc set env` pour définir les variables d'environnement de l'application à partir de secrets ou de mappages de configuration. Dans certains cas, les noms des clés peuvent être modifiés pour correspondre aux noms de variables d'environnement en utilisant l'option `--prefix`. Dans l'exemple suivant, la clé `user` définit la variable d'environnement `MYSQL_USER`, et la clé `root_password` définit la variable d'environnement `MYSQL_ROOT_PASSWORD`. Si le nom de la clé est indiqué en minuscules, la variable d'environnement correspondante est convertie en majuscules.

```
[user@host ~]$ oc set env deployment/demo --from secret/demo-secret \
>   --prefix MYSQL_
```

## Secrets sous la forme de fichiers dans un pod

Un secret peut être monté sur un répertoire au sein d'un pod. Un fichier est créé pour chaque clé du secret à l'aide du nom de la clé. Le contenu de chaque fichier est la valeur décodée du secret. La commande suivante montre comment monter des secrets dans un pod :

```
[user@host ~]$ oc set volume deployment/demo \ ①
>   --add --type secret \ ②
>   --secret-name demo-secret \ ③
>   --mount-path /app-secrets ④
```

- ① Modifiez la configuration du volume dans le déploiement demo.
- ② Ajoutez un nouveau volume à partir d'un secret. Les mappages de configuration peuvent également être montés en tant que volumes.
- ③ Utilisez le secret `demo-secret`.
- ④ Rendez les données secrètes disponibles dans le répertoire `/app-secrets` du pod. Le contenu du fichier `/app-secrets/user` est `demo-user`. Le contenu du fichier `/app-secrets/root_password` est `zT1KTgk`.

L'image de conteneur peut déterminer l'emplacement du point de montage et les noms de fichiers attendus. Par exemple, une image de conteneur qui exécute NGINX peut spécifier l'emplacement du certificat SSL et l'emplacement de la clé correspondante dans le fichier de configuration `/etc/nginx/nginx.conf`. Si les fichiers attendus ne sont pas trouvés, il se peut que l'exécution du conteneur échoue.



### Important

Si le point de montage existe déjà dans le pod, tous les fichiers existants au niveau de ce point de montage sont masqués par le secret monté. Les fichiers existants ne sont ni visibles ni accessibles.

## Présentation du mappage de configuration

À l'instar des secrets, les mappages de configuration dissocient les informations de configuration des images de conteneur. Contrairement aux secrets, les informations contenues dans les mappages de configuration ne nécessitent aucune protection. Vous pouvez utiliser les données d'un mappage de configuration pour définir des variables d'environnement dans l'image de conteneur, ou monter le mappage de configuration en tant que volume dans l'image de conteneur.

Il n'est pas nécessaire de recréer les images de conteneur lors de la modification d'un secret ou d'un mappage de configuration. Les nouveaux pods utilisent les secrets et les mappages de configuration mis à jour. Vous pouvez supprimer des pods à l'aide de secrets et de mappages de configuration plus anciens.

La syntaxe de création d'un mappage de configuration est très semblable à celle qui est utilisée pour créer un secret. Les paires clé-valeur peuvent être saisies sur la ligne de commande ou le contenu d'un fichier peut être utilisé comme valeur d'une clé spécifiée. La commande suivante montre comment créer un mappage de configuration :

```
[user@host ~]$ oc create configmap my-config \
>   --from-literal key1=config1 --from-literal key2=config2
```

## Mise à jour des secrets et des mappages de configuration

Les secrets et les mappages de configuration nécessitent parfois des mises à jour. Utilisez la commande `oc extract` pour vous assurer de disposer des données les plus récentes.

Enregistrez les données dans un répertoire spécifique à l'aide de l'option `--to`. Chaque clé du secret ou du mappage de configuration crée un fichier ayant le même nom qu'elle. Le contenu de chaque fichier est la valeur de la clé associée. Si vous exécutez la commande `oc extract` plusieurs fois, utilisez l'option `--confirm` pour écraser les fichiers existants.

```
[user@host ~]$ oc extract secret/htpasswd-ppk1q -n openshift-config \
>   --to /tmp/ --confirm
```

Après avoir mis à jour les fichiers enregistrés en local, utilisez la commande `oc set data` pour mettre à jour le secret ou le mappage de configuration. Pour chaque clé qui nécessite une mise à jour, indiquez le nom d'une clé et la valeur associée. Si un fichier contient la valeur, utilisez l'option `--from-file`.

Dans l'exemple `oc extract` précédent, le secret `htpasswd-ppk1q` contenait une seule clé nommée `htpasswd`. À l'aide de la commande `oc set data`, vous pouvez spécifier de manière explicite le nom de la clé `htpasswd` en utilisant l'option `--from-file htpasswd=/tmp/htpasswd`. Si le nom de la clé n'est pas spécifié, c'est le nom de fichier qui est utilisé.

```
[user@host ~]$ oc set data secret/htpasswd-ppk1q -n openshift-config \
>   --from-file /tmp/htpasswd
```



## Références

Pour plus d'informations sur les secrets, reportez-vous à la section *Understanding secrets* du chapitre *Working with pods* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#nodes-pods-secrets-about\\_nodes-pods-secrets](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#nodes-pods-secrets-about_nodes-pods-secrets)

Pour plus d'informations sur le chiffrement Etcd, reportez-vous au chapitre *Encrypting Etcd data* de la documentation *Security* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/security\\_and\\_compliance/index#encrypting-etcd](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/security_and_compliance/index#encrypting-etcd)

## ► Exercice guidé

# Gestion des informations sensibles à l'aide des secrets

Dans cet exercice, vous allez gérer des informations à l'aide de secrets.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Gérer les secrets et les utiliser pour initialiser les variables d'environnement dans les applications.
- Utiliser des secrets pour une application de base de données MySQL.
- Affecter des secrets à une application qui se connecte à une base de données MySQL.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et télécharge les fichiers de ressources nécessaires à cet exercice.

```
[student@workstation ~]$ lab authorization-secrets start
```

## Instructions

- 1. Connectez-vous au cluster OpenShift et créez le projet `authorization-secrets`.

1.1. Connectez-vous au cluster en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
>     https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

1.2. Créez le projet `authorization-secrets`.

```
[student@workstation ~]$ oc new-project authorization-secrets
Now using project "authorization-secrets" on server
"https://api.ocp4.example.com:6443".
...output omitted...
```

- 2. Créez un secret avec les informations d'identification et de connexion qui permettent d'accéder à une base de données MySQL.

```
[student@workstation ~]$ oc create secret generic mysql \
>   --from-literal user=myuser --from-literal password=redhat123 \
>   --from-literal database=test_secrets --from-literal hostname=mysql
secret/mysql created
```

- 3. Déployez une base de données et ajoutez le secret pour la configuration utilisateur et de la base de données.

- 3.1. Essayez de déployer un serveur de base de données éphémère. Cette opération doit échouer, car l'image MySQL a besoin de variables d'environnement pour sa configuration initiale. Les valeurs de ces variables ne peuvent pas être affectées à partir d'un secret à l'aide de la commande `oc new-app`.

```
[student@workstation ~]$ oc new-app --name mysql \
>   --image registry.redhat.io/rhel8/mysql-80:1
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "mysql" created
  deployment.apps "mysql" created
  service "mysql" created
--> Success
...output omitted...
```

- 3.2. Exécutez la commande `oc get pods` avec l'option `-w` pour récupérer l'état du déploiement en temps réel. Notez que le pod de base de données est en état d'échec. Appuyez sur `Ctrl+C` pour quitter la commande.

```
[student@workstation ~]$ oc get pods -w
NAME          READY   STATUS      RESTARTS   AGE
mysql-7b58b9b68b-ftp6j  0/1     CrashLoopBackOff  3 (46s ago)  106s
mysql-7b58b9b68b-ftp6j  1/1     Running     4 (53s ago)  113s
mysql-7b58b9b68b-ftp6j  0/1     Error       4 (54s ago)  114s
mysql-7b58b9b68b-ftp6j  0/1     CrashLoopBackOff  4 (16s ago)  2m9s
```



### Note

Cela peut prendre un certain temps pour que le pod atteigne l'état d'erreur.

- 3.3. Utilisez le secret `mysql` pour initialiser les variables d'environnement sur le déploiement `mysql`. Le déploiement a besoin des variables d'environnement `MYSQL_USER`, `MYSQL_PASSWORD` et `MYSQL_DATABASE` pour une initialisation réussie. Le secret dispose des clés `user`, `password` et `database` qui peuvent être affectées au déploiement en tant que variables d'environnement, en ajoutant le préfixe `MYSQL_`.

```
[student@workstation ~]$ oc set env deployment/mysql --from secret/mysql \
>   --prefix MYSQL_
deployment.apps/mysql updated
```

- 3.4. Pour montrer comment un secret peut être monté en tant que volume, montez le secret `mysql` dans le répertoire `/run/secrets/mysql` à l'intérieur du pod. Cette

étape montre uniquement comment monter un secret en tant que volume. Elle n'est pas nécessaire pour corriger le déploiement.

```
[student@workstation ~]$ oc set volume deployment/mysql --add --type secret \
>   --mount-path /run/secrets/mysql --secret-name mysql
info: Generated volume name: volume-nrh7r
deployment.apps/mysql volume updated
```

- 3.5. La modification du déploiement à l'aide de la commande `oc set env` ou `oc set volume` déclenche un nouveau déploiement d'application. Vérifiez que l'application `mysql` est déployée correctement après avoir ajouté les variables d'environnement.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
mysql-7cd7499d66-gm2rh   1/1     Running   0          21s
```

Prenez note du nom du pod à l'état `Running` ; vous en aurez besoin dans les étapes suivantes.

- ▶ 4. Vérifiez que la base de données s'authentifie désormais en utilisant les variables d'environnement initialisées à partir du secret `mysql`.

- 4.1. Ouvrez une session shell à distance sur le pod `mysql` à l'état `Running`.

```
[student@workstation ~]$ oc rsh mysql-7cd7499d66-gm2rh
sh-4.4$
```

- 4.2. Démarrez une session MySQL pour vérifier que les variables d'environnement initialisées par le secret `mysql` ont été utilisées pour configurer l'application `mysql`.

```
sh-4.4$ mysql -u myuser --password=redhat123 test_secrets -e 'show databases;'
mysql: [Warning] Using a password on the command line interface can be insecure.
+-----+
| Database      |
+-----+
| information_schema |
| test_secrets   |
+-----+
```

- 4.3. Listez les points de montage du pod contenant le modèle `mysql`. Notez que le point de montage est associé à un système de fichiers temporaire (`tmpfs`). Cela vaut pour tous les secrets qui sont montés en tant que volumes.

```
sh-4.4$ df -h | grep mysql
tmpfs            15G   16K   15G   1% /run/secrets/mysql
```

- 4.4. Examinez les fichiers montés au niveau du point de montage `/run/secrets/mysql`. Chaque fichier correspond à un nom de clé dans le secret, et le contenu de chaque fichier est la valeur de la clé associée.

```
sh-4.4$ for FILE in $(ls /run/secrets/mysql)
> do
> echo "${FILE}: $(cat /run/secrets/mysql/${FILE})"
> done
database: test_secrets
hostname: mysql
password: redhat123
user: myuser
```

- 4.5. Fermez la session shell à distance pour continuer depuis votre machine workstation.

```
sh-4.4$ exit
exit
[student@workstation ~]$
```

- 5. Créez une nouvelle application qui utilise la base de données MySQL.

- 5.1. Créez une application avec l'image redhattraining/famous-quotes de Quay.io.

```
[student@workstation ~]$ oc new-app --name quotes \
>   --image quay.io/redhattraining/famous-quotes:2.1
--> Found container image 7ff1a7b (19 months old) from quay.io for "quay.io/
redhattraining/famous-quotes:2.1"
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "quotes" created
  deployment.apps "quotes" created
  service "quotes" created
--> Success
...output omitted...
```

- 5.2. Vérifiez l'état du pod d'application quotes. Le pod affiche une erreur, car il ne peut pas se connecter à la base de données. Cela peut prendre un certain temps pour s'afficher dans la sortie. Appuyez sur Ctrl+C pour quitter la commande.

```
[student@workstation ~]$ oc get pods -l deployment=quotes -w
NAME                  READY   STATUS      RESTARTS   AGE
quotes-66b987df5d-59flf 0/1     CrashLoopBackOff 2 (20s ago) 50s
quotes-66b987df5d-59flf 0/1     Error        3          59s
quotes-66b987df5d-59flf 0/1     CrashLoopBackOff 3 (16s ago) 74s
```

- 6. L'application quotes nécessite plusieurs variables d'environnement. Le secret mysql peut initialiser des variables d'environnement pour l'application quotes en ajoutant le préfixe QUOTES\_.

- 6.1. Le secret mysql permet d'initialiser les variables d'environnement suivantes nécessaires à l'application quotes pour se connecter à la base de données : QUOTES\_USER, QUOTES\_PASSWORD, QUOTES\_DATABASE et QUOTES\_HOSTNAME, qui correspondent aux clés user, password, database et hostname du secret mysql.

```
[student@workstation ~]$ oc set env deployment/quotes --from secret/mysql \
>   --prefix QUOTES_
deployment.apps/quotes updated
```

- 6.2. Attendez que le pod d'application quotes soit redéployé. Il est mis fin automatiquement aux anciens pods.

```
[student@workstation ~]$ oc get pods -l deployment=quotes
NAME           READY   STATUS    RESTARTS   AGE
quotes-77df54758b-mqdtf   1/1     Running   3          7m17s
```

**Note**

Cela peut prendre un certain temps pour que le pod termine le déploiement.

- ▶ 7. Vérifiez que le pod quotes a réussi à se connecter à la base de données et que l'application affiche une liste de guillemets.
- 7.1. Examinez les journaux du pod à l'aide de la commande `oc logs`. Ces journaux indiquent une connexion réussie à la base de données.

```
[student@workstation ~]$ oc logs quotes-77df54758b-mqdtf | head -n2
... Connecting to the database: myuser:redhat123@tcp(mysql:3306)/test_secrets
... Database connection OK
```

- 7.2. Exposez le service quotes afin qu'il soit accessible depuis l'extérieur du cluster.

```
[student@workstation ~]$ oc expose service quotes \
>   --hostname quotes.apps.ocp4.example.com
route.route.openshift.io/quotes exposed
```

- 7.3. Vérifiez le nom d'hôte de l'application.

```
[student@workstation ~]$ oc get route quotes
NAME      HOST/PORT          PATH  SERVICES  PORT  ...
quotes   quotes.apps.ocp4.example.com      quotes   8000-tcp ...
```

- 7.4. Vérifiez que les variables sont correctement définies dans l'application en accédant au point d'entrée de l'API REST env.

```
[student@workstation ~]$ curl -s \
>   http://quotes.apps.ocp4.example.com/env | grep QUOTES_
<li>QUOTES_USER: myuser </li>
<li>QUOTES_PASSWORD: redhat123 </li>
<li>QUOTES_DATABASE: test_secrets</li>
<li>QUOTES_HOST: mysql</li>
```

- 7.5. Accédez au point d'entrée de l'API REST status de l'application pour tester la connexion de la base de données.

```
[student@workstation ~]$ curl -s http://quotes.apps.ocp4.example.com/status  
Database connection OK
```

- 7.6. Testez la fonctionnalité de l'application en accédant au point d'entrée de l'API REST random.

```
[student@workstation ~]$ curl -s http://quotes.apps.ocp4.example.com/random  
8: Those who can imagine anything, can create the impossible.  
- Alan Turing
```

- 8. Supprimez le projet authorization-secrets.

```
[student@workstation ~]$ oc delete project authorization-secrets  
project.project.openshift.io "authorization-secrets" deleted
```

## Fin

Sur la machine `workstation`, exéutez la commande `lab` pour mettre fin à l'exercice.

```
[student@workstation ~]$ lab authorization-secrets finish
```

L'exercice guidé est maintenant terminé.

# Contrôle des autorisations d'application à l'aide des contraintes du contexte de sécurité (SCC)

---

## Résultats

À la fin de cette section, vous devez pouvoir créer des comptes de service, appliquer des autorisations et gérer les contraintes de contexte de sécurité.

## Contraintes du contexte de sécurité (SCC)

Red Hat OpenShift fournit des *contraintes du contexte de sécurité* (SCC), un mécanisme de sécurité qui limite l'accès aux ressources, mais pas aux opérations dans OpenShift.

Les SCC limitent l'accès à l'environnement hôte d'un pod en cours d'exécution dans OpenShift.  
Les SCC contrôlent :

- l'exécution des conteneurs dotés de priviléges ;
- la demande de capacités supplémentaires à un conteneur ;
- l'utilisation de répertoires hôtes en tant que volumes ;
- la modification du contexte SELinux d'un conteneur ;
- la modification de l'ID utilisateur.

Certains conteneurs développés par la communauté peuvent nécessiter des contraintes du contexte de sécurité assouplies, afin d'accéder aux ressources qui sont interdites par défaut, telles que les systèmes de fichiers et les sockets, ou d'accéder à un contexte SELinux.

Vous pouvez exécuter la commande suivante en tant qu'administrateur de cluster pour répertorier les SCC définies par OpenShift :

```
[user@host ~]$ oc get scc
```

OpenShift fournit huit SCC par défaut :

- anyuid
- hostaccess
- hostmount-anyuid
- hostnetwork
- node-exporter
- nonroot
- privileged
- restricted

Pour obtenir plus d'informations sur une SCC, utilisez la commande `oc describe` :

```
[user@host ~]$ oc describe scc anyuid
Name:          anyuid
Priority:      10
Access:
  Users:        <none>
  Groups:       system:cluster-admins
Settings:
  ...output omitted...
```

Tous les pods créés par OpenShift utilisent la SCC nommée **restricted** qui fournit un accès limité aux ressources externes à OpenShift. Utilisez la commande `oc describe` pour afficher la contrainte de contexte de sécurité utilisée par un pod.

```
[user@host ~]$ oc describe pod console-5df4fcbb47-67c52 \
> -n openshift-console | grep scc
          openshift.io/scc: restricted
```

Lorsque la SCC **restricted** est utilisée, l'exécution des images de conteneur téléchargées à partir de registres de conteneur publics, tels que Docker Hub, risque d'échouer. Par exemple, une image de conteneur qui doit s'exécuter en tant qu'ID utilisateur spécifique peut échouer, car la SCC **restricted** exécute le conteneur à l'aide d'un ID utilisateur aléatoire. Une image de conteneur qui écoute sur le port 80 ou le port 443 peut échouer pour une raison semblable. L'ID utilisateur aléatoire utilisé par la SCC **restricted** ne peut pas démarrer un service qui écoute sur un port réseau avec des priviléges (numéros de port inférieurs à 1024). Utilisez la sous-commande `scc-subject-review` pour lister toutes les contraintes de contexte de sécurité permettant de surmonter les limitations d'un conteneur :

```
[user@host ~]$ oc get pod podname -o yaml | \
>   oc adm policy scc-subject-review -f -
```

Dans le cas de la SCC **anyuid**, la stratégie `run as user` est définie en tant que `RunAsAny`, ce qui signifie que le pod peut être exécuté comme n'importe quel ID utilisateur disponible dans le conteneur. Cette stratégie exige que les conteneurs qui nécessitent un utilisateur spécifique exécutent les commandes à l'aide d'un ID utilisateur précis.

Afin que le conteneur puisse s'exécuter avec une autre SCC, vous devez créer un compte de service lié à un pod. Utilisez la commande `oc create serviceaccount` pour créer le compte de service et utilisez l'option `-n` si le compte de service doit être créé dans un espace de noms différent de celui en cours :

```
[user@host ~]$ oc create serviceaccount service-account-name
```

Pour associer le compte de service à une SCC, utilisez la commande `oc adm policy`. Utilisez l'option `-z` pour identifier un compte de service et l'option `-n` si le compte de service existe dans un espace de noms différent de celui en cours :

```
[user@host ~]$ oc adm policy add-scc-to-user SCC -z service-account
```



### Important

L'affectation d'une SCC à un compte de service ou la suppression d'un SCC d'un compte de service doit être effectuée par un administrateur de cluster. Autoriser des pods à s'exécuter avec une SCC moins restrictive est susceptible de réduire la sécurité de votre cluster. Cette méthode doit être utilisée avec prudence.

Modifiez une configuration de déploiement ou un déploiement existant afin d'utiliser le compte de service à l'aide de la commande `oc set serviceaccount` :

```
[user@host ~]$ oc set serviceaccount deployment/deployment-name \
>     service-account-name
```

Si la commande réussit, les pods associés à la configuration de déploiement ou au déploiement sont redéployés.

## Conteneurs dotés de privilèges

Certains conteneurs peuvent avoir besoin d'accéder à l'environnement d'exécution de l'hôte. Par exemple, les conteneurs du créateur S2I sont une classe de conteneurs dotés de privilèges qui exigent un accès au-delà des limites de leurs propres conteneurs. Ces conteneurs peuvent présenter des risques de sécurité, car ils peuvent utiliser toute ressource d'un nœud OpenShift. Utilisez les SCC pour activer l'accès des conteneurs dotés de privilèges en créant des comptes de service dotés d'un accès avec privilèges.



### Références

Pour plus d'informations, reportez-vous au chapitre *Managing Security Context Constraints* de la documentation *Authentication and Authorization* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/security\\_and\\_compliance/index#encrypting-etcd](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/security_and_compliance/index#encrypting-etcd)

## ► Exercice guidé

# Contrôle des autorisations d'application à l'aide des contraintes du contexte de sécurité (SCC)

Dans cet exercice, vous allez déployer des applications qui nécessitent des pods dotés de permissions étendues.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Créer des comptes de service et leur affecter des contraintes de contexte de sécurité (SCC).
- Affecter un compte de service à une configuration de déploiement.
- Exécuter des applications qui ont besoin de priviléges root.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée certains utilisateurs HTPasswd nécessaires à cet exercice.

```
[student@workstation ~]$ lab authorization-scc start
```

## Instructions

- 1. Connectez-vous au cluster OpenShift et créez le projet `authorization-scc`.

- 1.1. Connectez-vous au cluster en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
> https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

- 1.2. Créez le projet `authorization-scc`.

```
[student@workstation ~]$ oc new-project authorization-scc
Now using project "authorization-scc" on server ...
...output omitted...
```

- 2. Déployez une application nommée `gitlab` à l'aide de l'image de conteneur située sur `quay.io/redhattraining/gitlab-ce:8.4.3-ce.0`. Cette image est une copie de

l'image de conteneur disponible sur `docker.io/gitlab/gitlab-ce:8.4.3-ce.0`. Vérifiez que le pod échoue parce que l'image de conteneur a besoin de privilèges `root`.

## 2.1. Déployez l'application gitlab

```
[student@workstation ~]$ oc new-app --name gitlab \
>   --image quay.io/redhattraining/gitlab-ce:8.4.3-ce.0
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "gitlab" created
  deployment.apps "gitlab" created
  service "gitlab" created
--> Success
...output omitted...
```

## 2.2. Déterminez si l'application a bien été déployée. Cela doit générer une erreur, car cette image a besoin de privilèges `root` pour se déployer correctement.

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS            RESTARTS   AGE
gitlab-d89cd88f8-jwqbp  0/1    ContainerCreating   0          19s
gitlab-d89cd88f8-jwqbp  1/1    Running           0          30s
gitlab-d89cd88f8-jwqbp  0/1    Error             0          36s
```



### Note

Cela peut prendre un certain temps pour que l'image atteigne l'état `Error`. Vous pouvez également voir l'état `CrashLoopBackOff` tout en vérifiant l'intégrité du pod.

## 2.3. Examinez les journaux d'application pour vérifier que l'échec est dû à des privilèges insuffisants.

```
[student@workstation ~]$ oc logs pod/gitlab-7d67db7875-gcsjl
...output omitted...
=====
Recipe Compile Error in /opt/gitlab/embedded/cookbooks/cache/cookbooks/gitlab/
recipes/default.rb
=====

Chef::Exceptions::InsufficientPermissions
-----
directory[/etc/gitlab] (gitlab::default line 26) had an error:
  Chef::Exceptions::InsufficientPermissions: Cannot create directory[/etc/gitlab]
  at /etc/gitlab due to insufficient permissions
...output omitted...
```

## 2.4. Connectez-vous en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
>   https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

- 2.5. Vérifiez si l'utilisation d'une SCC différente permet de résoudre le problème des autorisations.

```
[student@workstation ~]$ oc get pod/gitlab-7d67db7875-gcsjl -o yaml | \
>   oc adm policy scc-subject-review -f -
RESOURCE                      ALLOWED BY
Pod/gitlab-7d67db7875-gcsjl  anyuid
```

- 3. Créez un compte de service et affectez-lui la SCC anyuid.

- 3.1. Créez un compte de service nommé gitlab-sa.

```
[student@workstation ~]$ oc create sa gitlab-sa
serviceaccount/gitlab-sa created
```

- 3.2. Affectez la SCC anyuid au compte de service gitlab-sa.

```
[student@workstation ~]$ oc adm policy add-scc-to-user anyuid -z gitlab-sa
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:anyuid added: "gitlab-sa"
```

- 4. Modifiez l'application gitlab de sorte qu'elle utilise le compte de service que vous venez de créer. Vérifiez que le nouveau déploiement a été effectué correctement.

- 4.1. Connectez-vous en tant qu'utilisateur developer.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.
...output omitted...
```

- 4.2. Affectez le compte de service gitlab-sa au déploiement gitlab.

```
[student@workstation ~]$ oc set serviceaccount deployment/gitlab gitlab-sa
deployment.apps/gitlab serviceaccount updated
```

- 4.3. Vérifiez que le redéploiement gitlab a réussi. Il se peut que vous deviez exécuter la commande `oc get pods` à plusieurs reprises jusqu'à ce que vous trouviez un pod d'application en cours d'exécution.

```
[student@workstation ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
gitlab-86d6d65-zm2fd  1/1     Running   0          55s
```

- 5. Vérifiez que l'application gitlab fonctionne correctement.

- 5.1. Exposez l'application `gitlab`. Étant donné que le service `gitlab` écoute sur les ports 22, 80 et 443, vous devez utiliser l'option `--port`.

```
[student@workstation ~]$ oc expose service/gitlab --port 80 \
>   --hostname gitlab.apps.ocp4.example.com
route.route.openshift.io/gitlab exposed
```

- 5.2. Obtenez l'itinéraire exposé.

```
[student@workstation ~]$ oc get routes
NAME      HOST/PORT          PATH  SERVICES  PORT  ...
gitlab    gitlab.apps.ocp4.example.com  gitlab  80  ...
```

- 5.3. Vérifiez que l'application `gitlab` répond aux requêtes HTTP.

```
[student@workstation ~]$ curl -sL http://gitlab.apps.ocp4.example.com/ | \
>   grep '<title>' 
<title>Sign in · GitLab</title>
```

#### ▶ 6. Supprimez le projet `authorization-scc`.

```
[student@workstation ~]$ oc delete project authorization-scc
project.project.openshift.io "authorization-scc" deleted
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab authorization-scc finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Configuration de la sécurité des applications

Au cours de cet atelier, vous allez créer un secret afin de partager des informations de configuration sensibles et modifier un déploiement pour qu'il s'exécute avec des paramètres moins restrictifs.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Fournir des informations sensibles aux déploiements à l'aide de secrets.
- Autoriser les applications à s'exécuter dans un environnement moins restrictif à l'aide de contraintes de contexte de sécurité.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et que vous pouvez vous connecter au cluster.

```
[student@workstation ~]$ lab authorization-review start
```

## Instructions

1. En tant qu'utilisateur `developer`, créez le projet `authorization-review`. Toutes les tâches supplémentaires de cet exercice utilisent le projet `authorization-review`.
2. Créez un secret nommé `review-secret` que vous utiliserez avec la base de données MySQL et les applications WordPress. Le secret doit inclure trois paires clé-valeur : `user=wpuser`, `password=redhat123` et `database=wordpress`.
3. Déployez une application de base de données MySQL nommée `mysql` à l'aide de l'image de conteneur située dans `registry.redhat.io/rhel8/mysql-80:1`. Modifiez le déploiement `mysql` de manière à utiliser le secret `review-secret` en tant que variables d'environnement. Les variables d'environnement doivent utiliser le préfixe `MYSQL_`.
4. Déployez une application WordPress nommée `wordpress` à l'aide de l'image de conteneur située sur `quay.io/redhattraining/wordpress:5.7-php7.4-apache`. Une fois le déploiement effectué, modifiez le déploiement `wordpress` afin d'utiliser le secret `review-secret` en tant que variables d'environnement supplémentaires.

L'application WordPress nécessite la définition de plusieurs variables d'environnement lors de l'exécution de la commande `oc new-app` :

- `WORDPRESS_DB_HOST` avec la valeur `mysql`.
- `WORDPRESS_DB_NAME` avec la valeur `wordpress`.
- `WORDPRESS_TITLE` avec la valeur `auth-review`.
- `WORDPRESS_USER` avec la valeur `wpuser`.

- WORDPRESS\_PASSWORD avec la valeur redhat123.
- WORDPRESS\_EMAIL avec la valeur student@redhat.com.
- WORDPRESS\_URL avec la valeur wordpress-review.apps.ocp4.example.com.

Les variables d'environnement supplémentaires doivent utiliser le préfixe WORDPRESS\_DB\_.



### Note

Le pod wordpress ne s'exécutera pas correctement tant que vous n'aurez pas modifié le déploiement pour qu'il utilise une contrainte de contexte de sécurité moins restrictive.

5. En tant qu'utilisateur admin, identifiez une SCC moins restrictive qui permet une exécution correcte du déploiement wordpress. Créez un compte de service nommé wordpress-sa et attribuez-lui la SCC anyuid. Modifiez le déploiement wordpress pour qu'il utilise le compte de service wordpress-sa.
6. En tant qu'utilisateur developer, rendez le service wordpress accessible aux requêtes externes à l'aide du nom d'hôte wordpress-review.apps.ocp4.example.com. Accédez à la route à l'aide d'un navigateur Web et vérifiez que l'application WordPress affiche l'assistant d'installation.

## Évaluation

En tant qu'utilisateur student sur la machine workstation, utilisez la commande lab pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab authorization-review grade
```

## Fin

En tant qu'utilisateur student sur la machine workstation, utilisez la commande lab pour effectuer cet exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab authorization-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Configuration de la sécurité des applications

Au cours de cet atelier, vous allez créer un secret afin de partager des informations de configuration sensibles et modifier un déploiement pour qu'il s'exécute avec des paramètres moins restrictifs.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Fournir des informations sensibles aux déploiements à l'aide de secrets.
- Autoriser les applications à s'exécuter dans un environnement moins restrictif à l'aide de contraintes de contexte de sécurité.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et que vous pouvez vous connecter au cluster.

```
[student@workstation ~]$ lab authorization-review start
```

## Instructions

1. En tant qu'utilisateur `developer`, créez le projet `authorization-review`. Toutes les tâches supplémentaires de cet exercice utilisent le projet `authorization-review`.
  - 1.1. Connectez-vous au cluster en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
> https://api.ocp4.example.com:6443
Login successful.
...output omitted...
```

- 1.2. Créez le projet `authorization-review`.

```
[student@workstation ~]$ oc new-project authorization-review
Now using project "authorization-review" on server
"https://api.ocp4.example.com:6443".
...output omitted...
```

2. Créez un secret nommé `review-secret` que vous utiliserez avec la base de données MySQL et les applications WordPress. Le secret doit inclure trois paires clé-valeur : `user=wptester`, `password=redhat123` et `database=wordpress`.

## 2.1. Créez un secret nommé review-secret.

```
[student@workstation ~]$ oc create secret generic review-secret \
>   --from-literal user=wpuser --from-literal password=redhat123 \
>   --from-literal database=wordpress
secret/review-secret created
```

3. Déployez une application de base de données MySQL nommée mysql à l'aide de l'image de conteneur située dans registry.redhat.io/rhel8/mysql-80:1. Modifiez le déploiement mysql de manière à utiliser le secret review-secret en tant que variables d'environnement. Les variables d'environnement doivent utiliser le préfixe MYSQL\_.

## 3.1. Créez une application pour déployer un serveur de base de données mysql.

```
[student@workstation ~]$ oc new-app --name mysql \
>   --image registry.redhat.io/rhel8/mysql-80:1
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "mysql" created
  deployment.apps "mysql" created
  service "mysql" created
--> Success
...output omitted...
```

- 3.2. Utilisez le secret review-secret pour initialiser les variables d'environnement sur le déploiement mysql. L'option --prefix garantit que toutes les variables injectées dans le pod à partir du secret commencent par MYSQL\_.

```
[student@workstation ~]$ oc set env deployment/mysql --prefix MYSQL_ \
>   --from secret/review-secret
deployment.apps/mysql updated
```

## 3.3. Vérifiez que le pod mysql a été redéployé correctement.

```
[student@workstation ~]$ watch oc get pods
```

Appuyez sur Ctrl+C pour quitter la commande watch une fois que le pod mysql affiche 1/1 et Running.

```
Every 2.0s: oc get pods
...  
NAME          READY   STATUS    RESTARTS   AGE
mysql-f675b96f8-vspb9   1/1     Running   0          20s
```

**Note**

Cela peut prendre quelques minutes pour que le déploiement s'effectue correctement après avoir défini le secret.

4. Déployez une application WordPress nommée wordpress à l'aide de l'image de conteneur située sur quay.io/redhattraining/wordpress:5.7-php7.4-apache. Une fois le

déploiement effectué, modifiez le déploiement `wordpress` afin d'utiliser le secret `review-secret` en tant que variables d'environnement supplémentaires.

L'application Wordpress nécessite la définition de plusieurs variables d'environnement lors de l'exécution de la commande `oc new-app` :

- `WORDPRESS_DB_HOST` avec la valeur `mysql`.
- `WORDPRESS_DB_NAME` avec la valeur `wordpress`.
- `WORDPRESS_TITLE` avec la valeur `auth-review`.
- `WORDPRESS_USER` avec la valeur `wpuser`.
- `WORDPRESS_PASSWORD` avec la valeur `redhat123`.
- `WORDPRESS_EMAIL` avec la valeur `student@redhat.com`.
- `WORDPRESS_URL` avec la valeur `wordpress-review.apps.ocp4.example.com`.

Les variables d'environnement supplémentaires doivent utiliser le préfixe `WORDPRESS_DB_`.



### Note

Le pod `wordpress` ne s'exécutera pas correctement tant que vous n'aurez pas modifié le déploiement pour qu'il utilise une contrainte de contexte de sécurité moins restrictive.

#### 4.1. Déployez une application `wordpress`.

```
[student@workstation ~]$ oc new-app --name wordpress \
>   --image quay.io/redhattraining/wordpress:5.7-php7.4-apache \
>   -e WORDPRESS_DB_HOST=mysql \
>   -e WORDPRESS_DB_NAME=wordpress \
>   -e WORDPRESS_TITLE=auth-review \
>   -e WORDPRESS_USER=wpuser \
>   -e WORDPRESS_PASSWORD=redhat123 \
>   -e WORDPRESS_EMAIL=student@redhat.com \
>   -e WORDPRESS_URL=wordpress-review.apps.ocp4.example.com
...output omitted...
-> Creating resources ...
  imagestream.image.openshift.io "wordpress" created
  deployment.apps "wordpress" created
  service "wordpress" created
--> Success
...output omitted...
```

#### 4.2. Utilisez le secret `review-secret` pour initialiser les variables d'environnement sur le déploiement `wordpress`. L'option `--prefix` garantit que les variables injectées dans le pod à partir du secret commencent par `WORDPRESS_DB_`.

```
[student@workstation ~]$ oc set env deployment/wordpress \
>   --prefix WORDPRESS_DB_ --from secret/review-secret
deployment.apps/wordpress updated
```

#### 4.3. Vérifiez que le redéploiement du pod `wordpress` n'aboutit pas, même après avoir injecté des variables à partir du secret `review-secret`.

```
[student@workstation ~]$ watch oc get pods -l deployment=wordpress
```

Attendez jusqu'à une minute, puis appuyez sur **Ctrl+C** pour quitter la commande **watch**. Le pod **wordpress** redémarre continuellement. Chaque fois, le pod passe à l'état **Error** et ensuite **CrashLoopBackOff**.

```
Every 2.0s: oc get pods -l deployment=wordpress
...
NAME          READY   STATUS        RESTARTS   AGE
wordpress-68c49c9d4-wq46g   0/1    CrashLoopBackoff   5          4m30s
```

4.4. Recherchez des messages d'erreur dans les journaux du pod.

```
[student@workstation ~]$ oc logs wordpress-68c49c9d4-wq46g
...output omitted...
(13)Permission denied: AH00072: make_sock: could not bind to address [::]:80
(13)Permission denied: AH00072: make_sock: could not bind to address 0.0.0.0:80
no listening sockets available, shutting down
AH00015: Unable to open logs
```

Par défaut, OpenShift empêche les pods de démarrer des services qui écoutent sur les ports inférieurs à 1024.

5. En tant qu'utilisateur **admin**, identifiez une SCC moins restrictive qui permet une exécution correcte du déploiement **wordpress**. Créez un compte de service nommé **wordpress-sa** et attribuez-lui la SCC **anyuid**. Modifiez le déploiement **wordpress** pour qu'il utilise le compte de service **wordpress-sa**.

5.1. Connectez-vous en tant qu'utilisateur **admin**.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.
...output omitted...
```

5.2. Vérifiez si l'utilisation d'une SCC différente résout le problème des autorisations.

```
[student@workstation ~]$ oc get pod/wordpress-68c49c9d4-wq46g -o yaml \
>   | oc adm policy scc-subject-review -f -
RESOURCE           ALLOWED BY
Pod/wordpress-68c49c9d4-wq46g   anyuid
```



### Important

La commande **oc adm policy** doit être exécutée en tant qu'utilisateur **admin**.

5.3. Créez un compte de service nommé **wordpress-sa**.

```
[student@workstation ~]$ oc create serviceaccount wordpress-sa
serviceaccount/wordpress-sa created
```

5.4. Accordez la SCC **anyuid** au compte de service **wordpress-sa**. Si le pod **wordpress** s'exécute en tant qu'utilisateur **root**, OpenShift l'autorise à démarrer un service sur le port 80.

```
[student@workstation ~]$ oc adm policy add-scc-to-user anyuid -z wordpress-sa
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:anyuid added:
"wordpress-sa"
```

- 5.5. Configurez le déploiement `wordpress` pour utiliser le compte de service `wordpress-sa`.

```
[student@workstation ~]$ oc set serviceaccount deployment/wordpress \
>   wordpress-sa
deployment.apps/wordpress serviceaccount updated
```

- 5.6. Vérifiez que le pod `wordpress` a été redéployé avec succès après la configuration du compte de service.

```
[student@workstation ~]$ watch oc get pods -l deployment=wordpress
```

Appuyez sur `Ctrl+C` pour quitter la commande `watch` une fois que le pod `wordpress` affiche 1/1 et Running.

```
Every 2.0s: oc get pods -l deployment=wordpress
```

NAME	READY	STATUS	RESTARTS	AGE
wordpress-bcb5d97f6-mwljs	1/1	Running	0	21s

6. En tant qu'utilisateur `developer`, rendez le service `wordpress` accessible aux requêtes externes à l'aide du nom d'hôte `wordpress-review.apps.ocp4.example.com`. Accédez à la route à l'aide d'un navigateur Web et vérifiez que l'application WordPress affiche l'assistant d'installation.

- 6.1. Utilisez la commande `oc expose` pour créer une route vers l'application `wordpress`.

```
[student@workstation ~]$ oc expose service/wordpress \
>   --hostname wordpress-review.apps.ocp4.example.com
route.route.openshift.io/wordpress exposed
```

- 6.2. Utilisez un navigateur Web pour vérifier l'accès à l'URL `http://wordpress-review.apps.ocp4.example.com`. Lorsque vous déployez correctement l'application, un assistant d'installation s'affiche dans le navigateur.

Vous pouvez également utiliser la commande `curl` pour accéder directement à l'URL d'installation.

```
[student@workstation ~]$ curl -s \
>   http://wordpress-review.apps.ocp4.example.com/wp-admin/install.php \
>   | grep Installation
<title>WordPress &rsaquo; Installation</title>
```

## Évaluation

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab authorization-review grade
```

## Fin

En tant qu'utilisateur **student** sur la machine **workstation**, utilisez la commande **lab** pour effectuer cet exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab authorization-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Les ressources secrètes vous permettent de séparer les informations sensibles des pods d'application. Vous exposez des secrets à un pod d'application, en tant que variables d'environnement ou en tant que fichiers ordinaires.
- OpenShift utilise des contraintes de contexte de sécurité (SCC) pour définir les interactions autorisées entre les pods et les ressources du système. Par défaut, les pods fonctionnent sous le contexte `restricted`, ce qui limite l'accès aux ressources du nœud.



## chapitre 12

# Configuration de la mise en réseau OpenShift pour les applications

### Objectif

Résoudre les problèmes liés aux réseaux SDN (Software-Defined Networking) OpenShift et configurer les politiques réseau.

### Résultats

- Résoudre les problèmes liés aux réseaux SDN OpenShift à l'aide de l'interface de ligne de commande.
- Autoriser et protéger les connexions réseau aux applications au sein d'un cluster OpenShift.
- Limiter le trafic réseau entre les projets et les pods.

### Sections

- Résolution des problèmes liés aux réseaux SDN OpenShift (et exercice guidé)
- Exposition d'applications pour un accès externe (et exercice guidé)
- Configuration des politiques réseau (et exercice guidé)

### Atelier

Configuration de la mise en réseau OpenShift pour les applications

# Résolution des problèmes liés au réseau défini par logiciel OpenShift

---

## Résultats

À la fin de cette section, vous devez pouvoir résoudre les problèmes liés au réseau défini par logiciel OpenShift à l'aide de l'interface de ligne de commande.

## Présentation du réseau défini par logiciel OpenShift

OpenShift met en œuvre un SDN (réseau défini par logiciel) pour gérer l'infrastructure réseau du cluster et des applications utilisateur. Le SDN est un modèle de réseau qui vous permet de gérer les services réseau grâce à l'abstraction de plusieurs couches réseau. Il dissocie le logiciel qui gère le trafic, appelé le *plan de contrôle*, des mécanismes sous-jacents qui acheminent le trafic, appelé le *plan de données*. Parmi les nombreuses fonctions du SDN, les normes ouvertes permettent aux fournisseurs de proposer leurs solutions, une gestion centralisée, un routage dynamique et l'isolation des tenants.

Dans OpenShift Container Platform, le SDN répond aux cinq exigences suivantes :

- La gestion du trafic réseau et des ressources réseau par programme, afin que les équipes de l'organisation puissent décider comment exposer leurs applications.
- La gestion des communications entre les conteneurs qui s'exécutent dans le même projet.
- La gestion des communications entre les pods, qu'ils appartiennent à un même projet ou qu'ils s'exécutent dans des projets distincts.
- La gestion des communications réseau d'un pod vers un service.
- La gestion des communications réseau d'un réseau externe vers un service, ou des conteneurs vers des réseaux externes.

L'implémentation du SDN crée un modèle rétrocompatible, dans lequel les pods sont comparables aux machines virtuelles en termes d'allocation de port, de location d'adresses IP et de réservation.

## Le modèle de réseau OpenShift

CNI (Container Network Interface) est une interface commune entre le fournisseur réseau et l'exécution du conteneur. Elle est implémentée sous la forme de plug-ins réseau. L'interface CNI fournit la spécification nécessaire pour que les plug-ins configurent les interfaces réseau à l'intérieur des conteneurs. Les plug-ins écrits dans la spécification autorisent différents fournisseurs réseau à contrôler le réseau de clusters OpenShift.

Le SDN utilise des plug-ins CNI pour créer des espaces de noms Linux afin de partitionner l'utilisation des ressources et des processus sur les hôtes physiques et virtuels. Cette implémentation permet aux conteneurs à l'intérieur des pods de partager des ressources réseau, telles que des périphériques, des piles IP, des règles de pare-feu et des tables de routage. Le SDN alloue une adresse IP routable unique à chaque pod, de sorte que vous puissiez accéder au pod à partir de n'importe quel autre service du même réseau.

Voici quelques plug-ins CNI courants utilisés dans OpenShift :

- Réseau SDN OpenShift

- OVN-Kubernetes
- Kuryr

Dans OpenShift 4.10, OpenShift SDN et OVN-Kubernetes sont les fournisseurs réseau par défaut.

Le fournisseur réseau SDN OpenShift utilise Open vSwitch (OVS) pour connecter des pods sur le même nœud et la tunnellation VXLAN (Virtual Extensible LAN) pour connecter des nœuds. OVN-Kubernetes utilise Open Virtual Network (OVN) pour gérer le réseau de clusters. OVN étend OVS avec des abstractions de réseau virtuel. Kuryr fournit une mise en réseau via les services Red Hat OpenStack Platform Neutron et Octavia.

## Migration des applications existantes

La conception du SDN facilite la mise en conteneur de vos applications existantes, car vous n'avez pas besoin de modifier la façon dont les composants de l'application communiquent les uns avec les autres. Si votre application comprend de nombreux services qui communiquent par le biais de la pile TCP/UDP, cette approche fonctionne toujours, puisque les conteneurs d'un pod partagent la même pile réseau.

Le schéma ci-dessous montre comment tous les pods sont connectés à un réseau partagé :

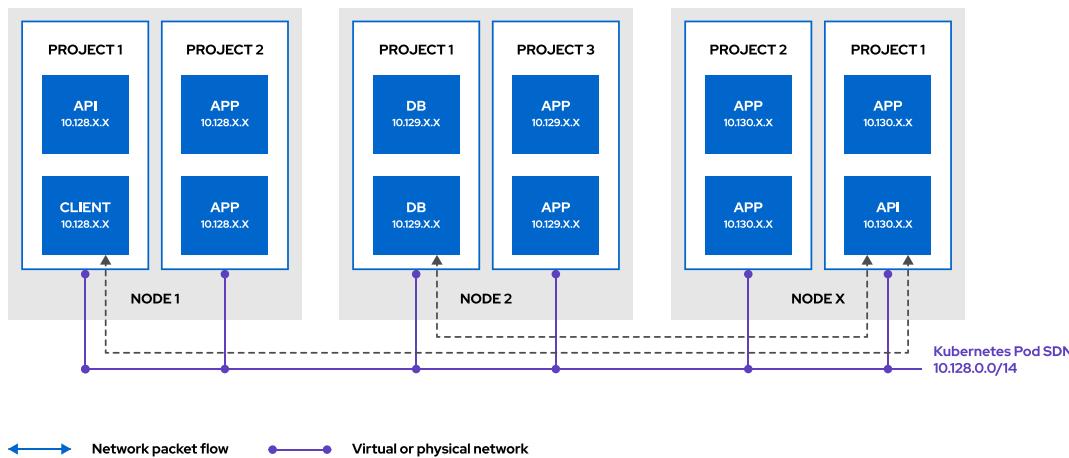


Figure 12.1: Réseau de base Kubernetes



### Note

L'opérateur de réseau du cluster OpenShift gère le réseau, comme nous l'expliquons plus loin.

## Utilisation des services pour accéder aux pods

Kubernetes fournit le concept de service, qui est une ressource essentielle dans toute application OpenShift. Les services permettent de regrouper logiquement les pods sous une route d'accès commune. Un service agit comme module d'équilibrage de charge devant un ou plusieurs pods, ce qui a pour effet de dissocier les spécifications de l'application (telles que le nombre de répliques en cours d'exécution) de l'accès à l'application. Il équilibre les charges des demandes des clients parmi les pods membres et fournit une interface stable qui permet la communication avec les pods sans avoir à effectuer le suivi des adresses IP des pods individuels.

La plupart des applications réelles ne s'exécutent pas comme un pod unique. Elles doivent évoluer horizontalement, de sorte qu'une application peut s'exécuter sur plusieurs pods pour répondre à

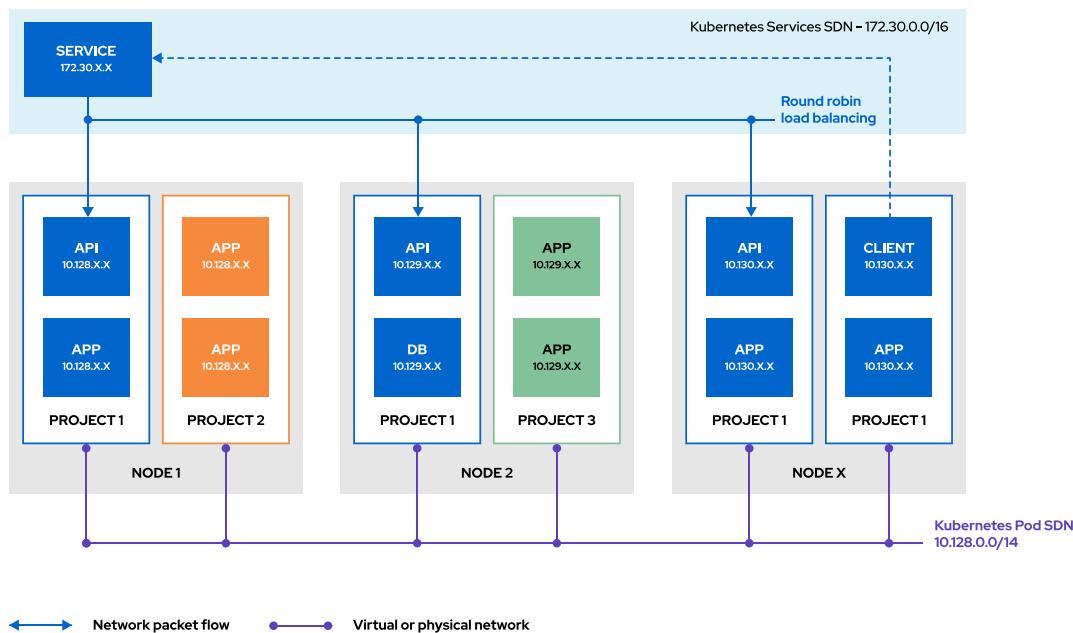
## chapitre 12 | Configuration de la mise en réseau OpenShift pour les applications

la demande croissante des utilisateurs. Dans un cluster OpenShift, les pods sont constamment créés et détruits sur les nœuds du cluster, par exemple lors du déploiement d'une nouvelle version de l'application ou lors du drainage d'un nœud pour la maintenance. Une adresse IP différente est attribuée aux pods chaque fois qu'ils sont créés ; par conséquent, les pods ne sont pas facilement adressables. Au lieu qu'un pod ait à découvrir l'adresse IP d'un autre pod, vous pouvez utiliser des services pour fournir une adresse IP unique que les autres pods peuvent utiliser, indépendamment de l'endroit où les pods s'exécutent.

Les services dépendent de sélecteurs (libellés) qui indiquent les pods qui reçoivent le trafic via le service. Chaque pod correspondant à ces sélecteurs est ajouté à la ressource de service en tant que point d'accès. Au fur et à mesure que les pods sont créés et supprimés, le service met automatiquement à jour les points d'accès. Les sélecteurs vous permettent de concevoir l'architecture et le routage de vos applications de manière plus souple. Par exemple, vous pouvez diviser l'application en niveaux et décider de créer un service pour chaque niveau. Les sélecteurs permettent d'avoir une conception flexible et hautement résiliente.

OpenShift utilise deux sous-réseaux : un pour les pods et un autre pour les services. Le trafic est acheminé de manière transparente vers les pods ; un agent (selon le mode réseau utilisé) gère les règles de routage pour acheminer le trafic vers les pods qui correspondent aux sélecteurs.

Le schéma ci-dessous montre comment trois pods d'API sont exécutés sur des nœuds séparés. Le service `service1` équilibre la charge entre ces trois pods.



**Figure 12.2: Utilisation des services pour accéder aux applications**

La définition YAML suivante montre comment créer un service. Elle définit le service `application-frontend`, ce qui crée une adresse IP virtuelle qui expose le port TCP 443. L'application frontale écoute sur le port non privilégié 8843.

```

kind: Service
apiVersion: v1
metadata:
  name: application-frontend ❶
  labels:
    app: frontend-svc ❷
  
```

```

spec:
  ports: ❸
    - name: HTTP ❹
      protocol: TCP
      port: 443 ❺
      targetPort: 8443 ❻
  selector: ❷
    app: shopping-cart
    name: frontend
  type: ClusterIP ❽

```

- ❶** Nom du service. Cet identifiant vous permet de gérer le service après sa création.
- ❷** Libellé que vous pouvez utiliser comme sélecteur. Il vous permet de regrouper de manière logique vos services.
- ❸** Ensemble d'objets décrivant les ports réseau à exposer.
- ❹** Chaque entrée définit le nom du mappage de port. Cette valeur est générique et est utilisée à des fins d'identification uniquement.
- ❺** Il s'agit du port exposé par le service. Vous utilisez ce port pour vous connecter à l'application exposée par le service.
- ❻** Il s'agit du port sur lequel l'application écoute. Le service crée une règle de transfert du port du service vers le port cible du service.
- ❼** Le sélecteur définit les pods qui se trouvent dans le pool de services. Les services utilisent ce sélecteur pour déterminer où acheminer le trafic. Dans cet exemple, le service cible tous les pods dont les libellés correspondent à `app: shopping-cart` et `name: frontend`.
- ❽** C'est de cette façon que le service est exposé. `ClusterIP` expose le service à l'aide d'une adresse IP interne au cluster et constitue la valeur par défaut. D'autres types de service seront décrits dans une autre partie de ce cours.

## L'opérateur DNS

L'opérateur DNS déploie et exécute un serveur DNS géré par CoreDNS, un serveur DNS allégé écrit en GoLang. L'opérateur DNS fournit une résolution de noms DNS entre les pods, ce qui permet aux services de découvrir leurs points d'accès.

Chaque fois que vous créez une nouvelle application, OpenShift configure les pods de manière à ce qu'ils contactent l'adresse IP du service CoreDNS pour la résolution DNS.

Exécutez la commande suivante pour vérifier la configuration de l'opérateur DNS :

```
[user@host ~]$ oc describe dns.operator/default
Name:          default
...output omitted...
API Version:  operator.openshift.io/v1
Kind:          DNS
...output omitted...
Spec:
Status:
```

```
Cluster Domain: cluster.local
Cluster IP: 172.30.0.10
...output omitted...
```

L'opérateur DNS est responsable des opérations suivantes :

- Créer un nom DNS de cluster par défaut (`cluster.local`)
- Affecter des noms DNS à des services que vous définissez (`db.backend.svc.cluster.local`, par exemple)

Par exemple, dans un pod nommé `example-6c4984d949-7m26r`, la commande suivante montre que vous pouvez atteindre le pod `hello` par le biais du service `hello` dans le projet `test` via le nom de domaine complet (FQDN) du service :

```
[user@host ~]$ oc rsh example-6c4984d949-7m26r \
> curl hello.test.svc.cluster.local:8080
```

## Gestion des enregistrements DNS pour les services

Cette implémentation de DNS permet aux pods de résoudre sans difficulté les noms DNS des ressources d'un projet ou du cluster. Les pods peuvent utiliser un schéma d'affectation de noms prévisible permettant d'accéder à un service. Par exemple, l'interrogation du nom d'hôte `db.backend.svc.cluster.local` à partir d'un conteneur renvoie l'adresse IP du service. Dans cet exemple, `db` est le nom du service, `backend` est le nom du projet et `cluster.local` est le nom DNS du cluster.

CoreDNS crée deux types d'enregistrements pour les services : les enregistrements A qui se résolvent en services et les enregistrements SRV qui correspondent au format suivant :

```
_port-name._port-protocol.svc-name.namespace.svc.cluster-domain.cluster-domain.
```

Par exemple, si vous utilisez un service qui expose le port TCP 443 par le biais du service HTTPS, l'enregistrement SRV est créé comme suit :

```
_443-tcp._tcp.https.frontend.svc.cluster.local.
```



### Note

Lorsque les services n'ont pas d'IP de cluster, l'opérateur DNS leur attribue un enregistrement DNS A qui se résout en un ensemble d'adresses IP des pods qui sous-tendent le service.

De la même manière, l'enregistrement SRV créé est résolu sur tous les pods qui sous-tendent le service.

## Présentation de l'opérateur de réseau du cluster

OpenShift Container Platform utilise l'opérateur de réseau du cluster pour gérer le SDN. Cela inclut le CIDR du réseau à utiliser, le fournisseur réseau et les pools d'adresses IP. La configuration de l'opérateur du réseau de clusters est effectuée avant l'installation, même s'il est possible de migrer du fournisseur réseau CNI par défaut OpenShift SDN vers le fournisseur réseau OVN-Kubernetes.

Exécutez la commande `oc get` en tant qu'utilisateur administratif pour consulter la configuration du SDN, gérée par la définition de ressources personnalisées `Network.config.openshift.io` :

```
[user@host ~]$ oc get network/cluster -o yaml
apiVersion: config.openshift.io/v1
kind: Network
...output omitted...
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14 ①
      hostPrefix: 23 ②
    externalIP:
      policy: {}
    networkType: OpenshiftSDN ③
    serviceNetwork:
      - 172.30.0.0/16
  ...output omitted...
```

- ① Définit le CIDR de tous les pods du cluster. Dans cet exemple, le masque de réseau du SDN est 255.252.0.0 ; le SDN peut allouer 262 144 adresses IP.
- ② Définit le préfixe d'hôte. Une valeur de 23 indique que le masque de réseau est 255.255.254.0, ce qui correspond à 512 adresses IP allouables.
- ③ Affiche le fournisseur SDN actuel. Vous pouvez choisir entre `OpenShiftSDN`, `OVNKubernetes` et `Kuryr`.

## Présentation du plug-in Multus CNI

Multus est un projet Open Source qui permet de prendre en charge plusieurs cartes réseau dans OpenShift. Multus résout notamment le problème de migration de la virtualisation de la fonction réseau vers des conteneurs. Multus fait office de courtier et d'arbitre pour les autres plug-ins CNI afin de gérer l'implémentation et le cycle de vie des périphériques réseau supplémentaires dans les conteneurs. Multus prend en charge des plug-ins tels que SR-IOV, vHost CNI, Flannel et Calico. Les cas limites spéciaux généralement observés dans les services de télécommunications, l'informatique en périphérie de réseau (Edge Computing) et la virtualisation sont gérés par Multus, ce qui permet de disposer de plusieurs interfaces réseau vers les pods.



### Note

Gardez à l'esprit que toutes les fonctions de mise en réseau Kubernetes et OpenShift, telles que les services, l'entrée et les routes, ignorent les périphériques réseau supplémentaires fournis par Multus.



## Références

Pour plus d'informations, reportez-vous au chapitre *Cluster Network Operator in OpenShift Container Platform* de la documentation *Networking* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/networking/index#cluster-network-operator](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/networking/index#cluster-network-operator)

### Réseau de clusters

<https://kubernetes.io/docs/concepts/cluster-administration/networking/>

### CoreDNS : DNS et découverte de services

<https://coredns.io/>

### CNI Multus

<https://github.com/intel/multus-cni>

## ► Exercice guidé

# Résolution des problèmes liés au réseau défini par logiciel OpenShift

Dans cet exercice, vous allez diagnostiquer et corriger les problèmes de connectivité avec un déploiement d'application de type Kubernetes.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Déployer l'application Node.js To Do.
- Créer un itinéraire pour exposer un service d'application.
- Résoudre les problèmes de communication entre les pods de votre application à l'aide de la commande `oc debug`.
- Mettre à jour un service OpenShift.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible.

```
[student@workstation ~]$ lab network-sdn start
```

## Instructions

En tant que développeur OpenShift, vous venez de terminer la migration d'une application Node.js To Do vers OpenShift. L'application comprend deux déploiements, l'un pour la base de données et l'autre pour l'application frontale. Elle contient également deux services pour la communication entre les pods.

Bien que l'application semble s'initialiser, vous ne pouvez pas y accéder via un navigateur Web. Dans cette activité, vous allez résoudre les problèmes liés à votre application et corriger le problème.

### ► 1. Connectez-vous au cluster OpenShift et créez le projet `network-sdn`.

- 1.1. Connectez-vous au cluster en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
>     https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Créez le projet `network-sdn`.

```
[student@workstation ~]$ oc new-project network-sdn
Now using project "network-sdn" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 2. Déployez la base de données et restaurez ses données. Le fichier ~/D0280/labs/network-sdn/todo-db.yaml définit les ressources suivantes :

- un déploiement qui crée un conteneur basé sur une image MySQL ;
- un service qui pointe vers l'application mysql.

2.1. Choisissez le répertoire ~/D0280/labs/network-sdn.

```
[student@workstation ~]$ cd ~/D0280/labs/network-sdn
```

- 2.2. Accédez au répertoire network-sdn et listez les fichiers. Dans une étape ultérieure, vous allez utiliser db-data.sql pour initialiser la base de données de l'application.

```
[student@workstation network-sdn]$ ls
db-data.sql  todo-db.yaml  todo-frontend.yaml
```

- 2.3. Utilisez oc create avec l'option -f sur todo-db.yaml pour déployer le pod du serveur de la base de données.

```
[student@workstation network-sdn]$ oc create -f todo-db.yaml
deployment.apps/mysql created
service/mysql created
```

- 2.4. Exécutez la commande oc status pour passer en revue les ressources présentes dans le projet. Le service mysql pointe vers le pod de la base de données.

```
[student@workstation network-sdn]$ oc status
In project network-sdn on server https://api.ocp4.example.com:6443

svc/mysql - 172.30.223.41:3306
  deployment/mysql deploys registry.redhat.io/rhel8/mysql-80:1
    deployment #1 running for 4 seconds - 0/1 pods
...output omitted...
```

- 2.5. Attendez quelques instants pour vous assurer que le pod de la base de données est en cours d'exécution. Récupérez le nom du pod de la base de données pour restaurer les tables de la base de données items.

```
[student@workstation network-sdn]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
mysql-94dc6645b-hjjqb   1/1     Running   0          33m
```

- 2.6. Utilisez la commande oc cp pour transférer le vidage de la base de données vers le pod. Veillez à remplacer le nom du pod par celui obtenu à l'étape précédente.

```
[student@workstation network-sdn]$ oc cp db-data.sql mysql-94dc6645b-hjjqb:/tmp/
```

- 2.7. Utilisez la commande `oc rsh` pour vous connecter au pod et restaurer la base de données.

```
[student@workstation network-sdn]$ oc rsh mysql-94dc6645b-hjjqb /bin/bash
bash-4.4$ mysql -u root items < /tmp/db-data.sql
```

- 2.8. Assurez-vous que la table `Item` figure bien dans la base de données.

```
bash-4.4$ mysql -u root items -e "SHOW TABLES;"
```

Tables_in_items	
Item	
+-----+	

- 2.9. Quittez le conteneur.

```
bash-4.4$ exit
exit
```

- 3. Déployez l'application frontale. Le fichier `~/D0280/labs/network-sdn/todo-frontend.yaml` définit les ressources suivantes :

- un déploiement qui crée l'application Node.js Todo ;
- un service qui pointe vers l'application `frontend`.

- 3.1. Utilisez la commande `oc create` pour créer l'application frontale.

```
[student@workstation network-sdn]$ oc create -f todo-frontend.yaml
deployment.apps/frontend created
service/frontend created
```

- 3.2. Attendez quelques instants que le conteneur frontal démarre, puis exécutez la commande `oc get pods`.

```
[student@workstation network-sdn]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
frontend-57b8b445df-f56qh   1/1     Running   0          34s
...output omitted...
```

- 4. Créez une route pour accéder au service `frontend` et accédez à l'application.

- 4.1. Vous devez créer un itinéraire pour accéder à l'application à partir d'un réseau externe. Pour créer cette route, utilisez la commande `oc expose` sur le service `frontend`. Utilisez l'option `--hostname` pour remplacer le nom de domaine complet par défaut créé par OpenShift.

```
[student@workstation network-sdn]$ oc expose service frontend \
>   --hostname todo.apps.ocp4.example.com
route.route.openshift.io/frontend exposed
```

- 4.2. Listez les itinéraires du projet.

```
[student@workstation network-sdn]$ oc get routes
NAME      HOST/PORT          PATH  SERVICES  PORT  ...
frontend  todo.apps.ocp4.example.com  frontend  8080  ...
```

Comme vous pouvez le voir dans cet exemple, OpenShift détecte le port sur lequel l'application écoute et crée une règle de transfert du port 80 vers le port 8080, qui est le port *cible*.

- 4.3. À partir de **workstation**, ouvrez Firefox et accédez à l'URL. Assurez-vous d'ajouter la barre oblique à la fin.

- <http://todo.apps.ocp4.example.com/todo/>

L'application n'est pas accessible, comme illustré dans la capture d'écran ci-dessous.

## Application is not available

The application is currently not serving requests at this endpoint. It may not have been started or is still starting.

**i** Possible reasons you are seeing this page:

- **The host doesn't exist.** Make sure the hostname was typed correctly and that a route matching this hostname exists.
- **The host exists, but doesn't have a matching path.** Check if the URL path was typed correctly and that the route was created using the desired path.
- **Route and path matches, but all pods are down.** Make sure that the resources exposed by this route (pods, services, deployment configs, etc) have at least one pod running.

- 4.4. Inspectez les journaux du pod à la recherche d'erreurs. La sortie n'indique aucune erreur.

```
[student@workstation network-sdn]$ oc logs frontend-57b8b445df-f56qh
App is ready at : 8080
```

- 5. Exécutez `oc debug` pour créer une copie carbone d'un pod existant dans le déploiement `frontend`. Vous utilisez ce pod pour vérifier la connectivité à la base de données.

- 5.1. Avant de créer un pod de débogage, récupérez l'adresse IP du service de base de données. Dans une étape ultérieure, vous allez utiliser la commande `curl` pour accéder au point d'accès de la base de données.

L'expression JSONPath vous permet de récupérer l'adresse IP du service.

```
[student@workstation network-sdn]$ oc get service/mysql \
>   -o jsonpath=".spec.clusterIP\n"
172.30.103.29
```

- 5.2. Exécutez la commande `oc debug` sur le déploiement `frontend`, qui exécute le pod d'application Web.

```
[student@workstation network-sdn]$ oc debug -t deployment/frontend
Starting pod/frontend-debug ...
Pod IP: 10.131.0.144
If you don't see a command prompt, try pressing enter.
sh-4.4$
```

- 5.3. Vous pouvez tester la connectivité entre le déploiement `frontend` et la base de données en utilisant notamment la commande `curl` qui prend en charge un grand nombre de protocoles.

Utilisez `curl` pour vous connecter à la base de données sur le port 3306, qui est le port MySQL par défaut. Veillez à remplacer l'adresse IP par celle obtenue précédemment pour le service `mysql`. Lorsque vous avez terminé, appuyez sur `Ctrl+C` pour quitter la session, puis saisissez `exit` pour quitter le pod de débogage.

```
sh-4.4$ curl -v telnet://172.30.103.29:3306
* About to connect() to 172.30.103.29 port 3306 (#0)
*   Trying 172.30.103.29...
* Connected to 172.30.103.29 (172.30.103.29) port 3306 (#0)
J
8.0.21
* RCVD IAC 2
* RCVD IAC 199
^C
sh-4.4$ exit
exit
```

Removing debug pod ...

La sortie indique que la base de données est en cours d'exécution et qu'elle est accessible depuis le déploiement `frontend`.

- 6. Dans les étapes suivantes, vous vous assurez que la connectivité réseau à l'intérieur du cluster est opérationnelle en vous connectant au conteneur frontal à partir du conteneur de la base de données.
- Récupérez des informations sur le pod `frontend`, puis utilisez la commande `oc debug` pour diagnostiquer le problème à partir du déploiement `mysql`.
- 6.1. Avant de créer un pod de débogage, récupérez l'adresse IP du service `frontend`.

```
[student@workstation network-sdn]$ oc get service/frontend \
>   -o jsonpath=".spec.clusterIP\{'\n'\}'"
172.30.23.147
```

- 6.2. Exécutez la commande `oc debug` pour créer un conteneur de résolution de problèmes en fonction du déploiement `mysql`. Vous devez remplacer l'image du conteneur, car l'image du serveur MySQL ne fournit pas la commande `curl`.

```
[student@workstation network-sdn]$ oc debug -t deployment/mysql \
>   --image registry.access.redhat.com/ubi8/ubi:8.4
Starting pod/mysql-debug ...
Pod IP: 10.131.0.146
If you don't see a command prompt, try pressing enter.
sh-4.4$
```

- 6.3. Utilisez la commande `curl` pour vous connecter à l'application `frontend` sur le port 8080. Veillez à remplacer l'adresse IP par celle obtenue précédemment pour le service `frontend`.

La sortie ci-dessous indique que le temps d'attente de la commande `curl` est dépassé. Cela peut indiquer que l'application n'est pas en cours d'exécution ou que le service n'est pas en mesure d'accéder à l'application.

```
sh-4.4$ curl -m 10 -v http://172.30.23.147:8080
* Rebuilt URL to: http://172.30.23.147:8080/
*   Trying 172.30.23.147...
* TCP_NODELAY set
* Connection timed out after 10000 milliseconds
* Closing connection 0
curl: (28) Connection timed out after 10000 milliseconds
```

- 6.4. Quittez le pod de débogage.

```
sh-4.4$ exit
exit

Removing debug pod ...
```

- 7. Dans les étapes suivantes, vous vous connectez au pod `frontend` via son adresse IP privée. Cela permet de vérifier si le problème est lié au service.

- 7.1. Récupérez l'adresse IP du pod `frontend`.

```
[student@workstation network-sdn]$ oc get pods -o wide -l name=frontend
NAME                  READY   STATUS    RESTARTS   AGE     IP           ...
frontend-57b8b445df-f56qh   1/1     Running   0          39m   10.128.2.61  ...
```

- 7.2. Créez un pod de débogage à partir du déploiement `mysql`.

```
[student@workstation network-sdn]$ oc debug -t deployment/mysql \
>   --image registry.access.redhat.com/ubi8/ubi:8.4
Starting pod/mysql-debug ...
Pod IP: 10.131.1.27
If you don't see a command prompt, try pressing enter.
sh-4.4$
```

- 7.3. Exécutez la commande `curl` en mode détaillé sur le pod `frontend`, sur le port 8080. Remplacez l'adresse IP par celle obtenue précédemment pour le pod `frontend`.

```
sh-4.4$ curl -v http://10.128.2.61:8080/todo/
*   Trying 10.128.2.61...
* TCP_NODELAY set
* Connected to 10.128.2.61 (10.128.2.61) port 8080 (#0)
> GET /todo/ HTTP/1.1
> Host: 10.128.2.61:8080
> User-Agent: curl/7.61.1
> Accept: /
>
< HTTP/1.1 200 OK
...output omitted...
```

La commande `curl` peut accéder à l'application via l'adresse IP privée du pod.

#### 7.4. Quittez le pod de débogage.

```
sh-4.2$ exit
exit

Removing debug pod ...
```

### ► 8. Examinez la configuration du service frontend.

#### 8.1. Listez les services du projet et assurez-vous que le service `frontend` existe.

```
[student@workstation network-sdn]$ oc get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
frontend  ClusterIP  172.30.23.147  <none>          8080/TCP    93m
mysql     ClusterIP  172.30.103.29   <none>          3306/TCP    93m
```

#### 8.2. Examinez la configuration et l'état du service `frontend`. Notez la valeur du sélecteur de service qui indique le pod auquel le service doit transférer les paquets.

```
[student@workstation network-sdn]$ oc describe svc/frontend
Name:            frontend
Namespace:       network-sdn
Labels:          app=todonodejs
                 name=frontend
Annotations:    <none>
Selector:        name=api
Type:            ClusterIP
IP:              172.30.23.147
Port:            <unset>  8080/TCP
TargetPort:      8080/TCP
Endpoints:      <none>
Session Affinity: None
Events:          <none>
```

Cette sortie indique également que le service n'a pas de point d'accès ; il ne peut donc pas transférer le trafic entrant vers l'application.

- 8.3. Récupérez les libellés du déploiement **frontend**. La sortie montre que les pods sont créés avec un libellé **name** dont la valeur est **frontend**, tandis que le service de l'étape précédente utilise **api** comme valeur.

```
[student@workstation network-sdn]$ oc describe deployment/frontend | \
>   grep -A1 Labels
Labels:           app=todonodejs
                  name=frontend
--
Labels:  app=todonodejs
        name=frontend
```

► **9.** Mettez à jour le service **frontend** et accédez à l'application.

- 9.1. Exécutez la commande **oc edit** pour modifier le service **frontend**. Mettez à jour le sélecteur pour qu'il corresponde au bon libellé.

```
[student@workstation network-sdn]$ oc edit svc/frontend
```

Localisez la section qui définit le sélecteur, puis mettez à jour le libellé **name** : **frontend** à l'intérieur du sélecteur. Après avoir effectué les modifications, quittez l'éditeur.

```
...output omitted...
selector:
  name: frontend
...output omitted...
```

Enregistrez vos modifications et vérifiez que la commande **oc edit** les a appliquées.

```
service/frontend edited
```

- 9.2. Vérifiez la configuration du service pour vous assurer que le service dispose d'un point d'accès.

```
[student@workstation network-sdn]$ oc describe svc/frontend
Name:           frontend
Namespace:      network-sdn
Labels:         app=todonodejs
                name=frontend
Annotations:    <none>
Selector:    name=frontend
Type:          ClusterIP
IP:            172.30.169.113
Port:          <unset>  8080/TCP
TargetPort:    8080/TCP
Endpoints:   10.128.2.61:8080
Session Affinity: None
Events:        <none>
```

- 9.3. Sur la machine **workstation**, ouvrez Firefox et accédez à l'application To Do, à l'adresse suivante :

- <http://todo.apps.ocp4.example.com/todo/>

Vous devez voir l'application To Do.

Id	Description	Done	
1	Pick up newsp...	false	
2	Buy groceries	true	

First Previous **1** Next Last

**Add Task**

Description:

Completed:

**Clear** **Save**

#### ► 10. Effectuez un nettoyage.

10.1. Ouvrez le répertoire personnel de l'utilisateur.

```
[student@workstation network-sdn]$ cd
```

10.2. Supprimez le projet network-sdn.

```
[student@workstation ~]$ oc delete project network-sdn
project.project.openshift.io "network-sdn" deleted
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab network-sdn finish
```

L'exercice guidé est maintenant terminé.

# Rendre les applications accessibles en externe

## Résultats

Après avoir terminé cette section, vous devez pouvoir autoriser et protéger les connexions réseau aux applications au sein d'un cluster OpenShift.

## Accès à l'application à partir de réseaux externes

OpenShift Container Platform offre de nombreuses méthodes pour exposer vos applications à des réseaux externes. Vous pouvez exposer le trafic HTTP et HTTPS, les applications TCP, mais également le trafic non TCP. Certaines de ces méthodes sont des types de service, tels que NodePort ou le module d'équilibrage de charge, tandis que d'autres utilisent leur propre ressource d'API, comme Ingress et Route.

Les *itinéraires* OpenShift vous permettent d'exposer vos applications à des réseaux externes. Avec les itinéraires, vous pouvez accéder à votre application à l'aide d'un nom d'hôte unique accessible au public. Les itinéraires s'appuient sur un *plug-in* de routeur pour rediriger le trafic de l'adresse IP publique vers les pods.

Le schéma ci-dessous montre comment une route expose une application qui s'exécute sous la forme de pods dans votre cluster :

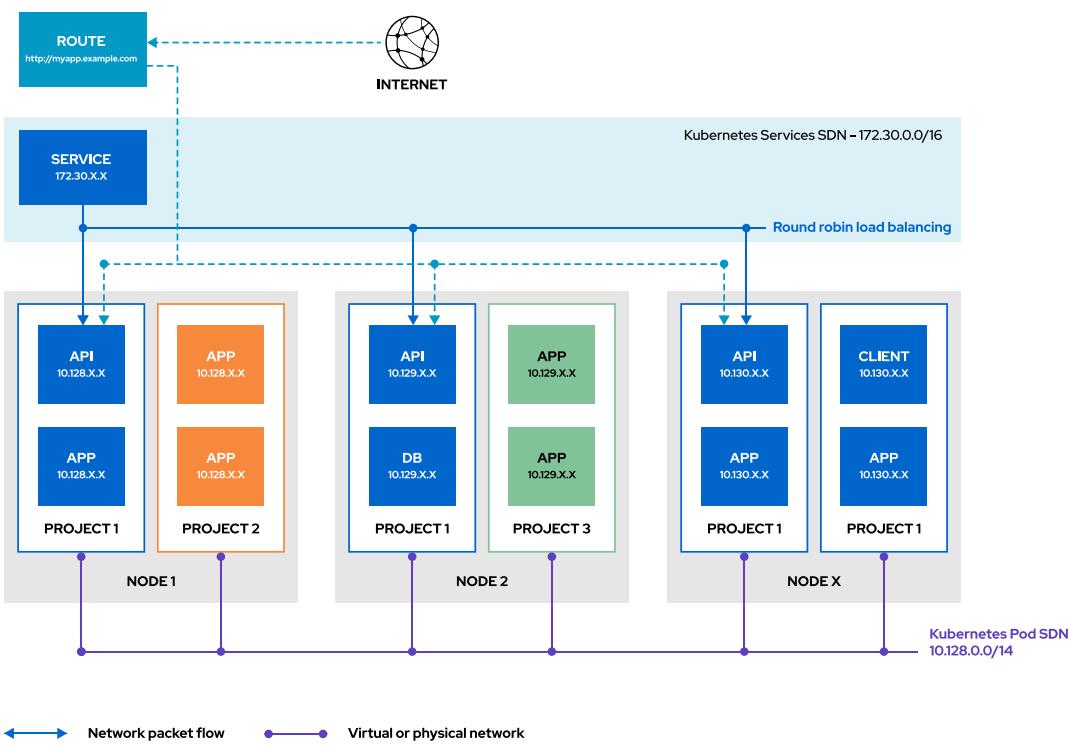


Figure 12.5: Utilisation de routes pour exposer des applications

**Note**

Pour des raisons de performances, les routeurs envoient des demandes directement aux pods en fonction de la configuration du service.

La ligne en pointillés indique cette implémentation. En d'autres termes, le routeur accède aux pods par le biais du réseau des services.

## Description des méthodes de gestion du trafic entrant

Le contrôleur d'entrée (Ingress Controller) constitue le moyen le plus courant de gérer le trafic entrant. OpenShift implémente ce contrôleur avec un service de routeur partagé qui s'exécute sous la forme d'un pod à l'intérieur du cluster. Vous pouvez mettre à l'échelle et répliquer ce pod comme tout autre pod standard. Ce service de routage est basé sur le logiciel Open Source HAProxy.

Les routes et l'entrée constituent les ressources principales pour la gestion du trafic entrant :

### **Route**

Les routes transmettent le trafic entrant aux services dans le cluster. Elles ont été créées avant les objets entrants Kubernetes et fournissent davantage de fonctions. Les routes fournissent des fonctions avancées qui peuvent ne pas être prises en charge par les contrôleurs d'entrée Kubernetes via une interface standard, telle que le rechiffrement TLS, le relais TLS et le trafic partagé pour les déploiements bleu-vert.

### **Ingress**

Une entrée est une ressource Kubernetes qui fournit les mêmes fonctions que les routes (qui constituent une ressource OpenShift). Les entrées acceptent les demandes externes et les transmettent par proxy en fonction de la route. Vous ne pouvez autoriser que certains types de trafic : HTTP, HTTPS, SNI (*Identification de nom de serveur*) et TLS avec SNI. Dans OpenShift, les routes sont générées pour répondre aux conditions spécifiées par l'objet d'entrée.

Il existe des alternatives à l'entrée et aux routes, mais elles sont réservées à des cas d'utilisation spéciaux. Les types de service suivants permettent un accès externe aux services :

### **External load balancer**

Cette ressource indique à OpenShift de faire tourner un module d'équilibrage de charge dans un environnement cloud. Un module d'équilibrage de charge indique à OpenShift d'interagir avec le fournisseur de cloud dans lequel s'exécute le cluster pour déployer un module de ce type.

### **Service external IP**

Cette méthode indique à OpenShift de définir des règles NAT pour rediriger le trafic de l'une des adresses IP du cluster vers le conteneur.

### **NodePort**

Avec cette méthode, OpenShift expose un service sur un port statique sur l'adresse IP du nœud. Vous devez vous assurer que les adresses IP externes sont correctement acheminées vers les nœuds.

## Création de routes

La méthode la plus simple et privilégiée pour créer une route (sécurisée ou non) consiste à utiliser la commande `oc expose service service`, où `service` correspond à un service. Utilisez l'option `--hostname` pour fournir un nom d'hôte personnalisé pour la route.

```
[user@host ~]$ oc expose service api-frontend \
>   --hostname api.apps.acme.com
```

Lorsque vous omettez le nom d'hôte, OpenShift en génère un pour vous avec la structure suivante : <nom-route>-<nom-projet>. <domaine-par-défaut>. Par exemple, si vous créez une route **frontend** dans un projet **api**, dans un cluster qui utilise **apps.example.com** comme domaine générique, le nom d'hôte de la route sera le suivant :

```
frontend-api.apps.example.com.
```



### Important

Le serveur DNS qui héberge le domaine générique ignore tous les noms d'hôte de l'itinéraire ; il ne résout que les noms avec les adresses IP configurées. Seul le routeur OpenShift connaît les noms d'hôte de la route, traitant chacun comme un hôte virtuel HTTP.

Les noms d'hôtes de domaines génériques non valides, c'est-à-dire les noms d'hôtes qui ne correspondent à aucune route, sont bloqués par le routeur OpenShift et entraînent une erreur HTTP 404.

Lors de la création d'une route, veuillez prendre en compte les paramètres suivants :

- Le nom d'un service. L'itinéraire utilise le service pour déterminer les pods vers lesquels acheminer le trafic.
- Un nom d'hôte pour l'itinéraire. Un itinéraire est toujours un sous-domaine de votre domaine générique de cluster. Par exemple, si vous utilisez un domaine générique **dapps.dev-cluster.acme.com** et que vous devez exposer un service frontal par le biais d'un itinéraire, il aura le nom suivant :

```
frontend.apps.dev-cluster.acme.com.
```



### Note

Vous pouvez également laisser OpenShift générer automatiquement un nom d'hôte pour l'itinéraire.

- Un chemin facultatif, pour les routes basées sur des chemins.
- Un port cible sur lequel l'application écoute. Le port cible correspond généralement au port que vous définissez dans la clé **targetPort** d'un service.
- Une stratégie de chiffrement, selon que vous avez besoin d'un itinéraire sécurisé ou non sécurisé.

La liste suivante présente une définition minimale pour un itinéraire :

```
kind: Route
apiVersion: route.openshift.io/v1
metadata:
  name: a-simple-route ①
```

```
labels: ②
  app: API
  name: api-frontend
spec:
  host: api.apps.acme.com ③
  to:
    kind: Service
    name: api-frontend ④
  port: ⑤
    targetPort: 8443
```

- ①** Nom de la route. Ce nom doit être unique.
- ②** Ensemble de libellés que vous pouvez utiliser comme sélecteurs.
- ③** Nom d'hôte de la route. Ce nom d'hôte doit être un sous-domaine de votre domaine générique, étant donné qu'OpenShift achemine le domaine générique vers les routeurs.
- ④** Service vers lequel rediriger le trafic. Bien que vous utilisiez un nom de service, l'itinéraire utilise uniquement ces informations pour déterminer la liste des pods qui reçoivent le trafic.
- ⑤** Port de l'application. Étant donné que les routes contournent les services, il doit correspondre au port d'application et non au port de service.

## Sécurisation des routes

Les routes peuvent être sécurisées ou non. Les routes sécurisées permettent d'utiliser plusieurs types de terminaisons TLS pour fournir des certificats au client. Les itinéraires non sécurisés sont les plus simples à configurer, car ils ne nécessitent aucune clé ni aucun certificat, mais les itinéraires sécurisés chiffrent le trafic vers et depuis les pods.

Une route sécurisée spécifie la terminaison TLS de la route. Les types de terminaisons disponibles sont présentés dans la liste suivante :

### Itinéraires sécurisés OpenShift

#### Edge

Avec la terminaison de périphérie, la terminaison TLS se produit au niveau du routeur, avant que le trafic ne soit acheminé vers les pods. Les certificats TLS sont fournis par le routeur, c'est pourquoi vous devez les configurer dans l'itinéraire ; sinon, OpenShift attribue son propre certificat au routeur pour la terminaison TLS. Étant donné que TLS prend fin au niveau du routeur, les connexions du routeur aux points d'accès sur le réseau interne ne sont pas chiffrées.

#### Transfert direct

Avec la terminaison directe, le trafic chiffré est envoyé directement au pod de destination sans que le routeur ne fournit la terminaison TLS. Dans ce mode, l'application est responsable de la fourniture de certificats pour le trafic. Actuellement, il s'agit de la seule méthode qui permet une authentification mutuelle entre l'application et un client qui y accède.

#### Rechiffrement

Le rechiffrement est une variation de la terminaison de périphérie, par laquelle le routeur met fin à TLS avec un certificat, puis chiffre à nouveau sa connexion au point d'accès, qui peut avoir un certificat différent. Par conséquent, le chemin complet de la connexion est chiffré, même sur le réseau interne. Le routeur utilise des bilans de santé pour déterminer l'authenticité de l'hôte.

## Sécurisation des applications avec des routes de périphérie

Avant de créer un itinéraire sécurisé, vous devez générer un certificat TLS. La commande suivante montre comment créer un itinéraire de périphérie sécurisé avec un certificat TLS :

```
[user@host ~]$ oc create route edge \
>   --service api-frontend --hostname api.apps.acme.com \
>   --key api.key --cert api.crt ① ②
```

- ① L'option `--key` requiert la clé privée du certificat.
- ② L'option `--cert` requiert le certificat qui a été signé avec cette clé.

Lors de l'utilisation d'un itinéraire en mode périphérie, le trafic entre le client et le routeur est chiffré, mais pas celui entre le routeur et l'application :

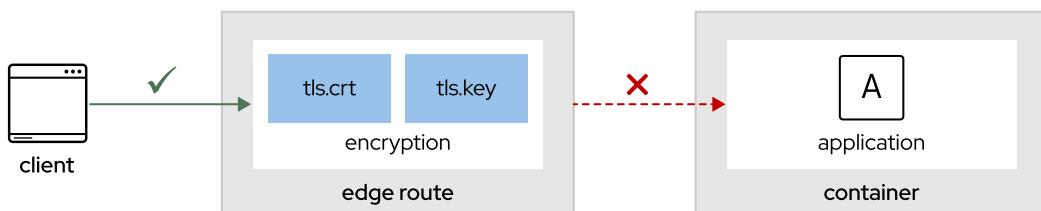


Figure 12.6: Sécurisation des applications avec des itinéraires de périphérie



### Note

Les politiques réseau peuvent vous aider à protéger le trafic interne entre vos applications ou entre les projets.

## Sécurisation des applications avec des routes directes

L'exemple précédent montre comment créer une route de périphérie, c'est-à-dire une route OpenShift qui présente un certificat à la périphérie. Les routes directes offrent une alternative sécurisée, car l'application expose son certificat TLS. Le trafic est ainsi chiffré entre le client et l'application.

Pour créer une route directe, vous avez besoin d'un certificat et d'un moyen qui permet à votre application d'y accéder. Le meilleur moyen d'y parvenir consiste à utiliser des secrets TLS OpenShift. Les secrets sont exposés par le biais d'un point de montage dans le conteneur.

Le schéma ci-dessous montre comment vous pouvez monter une ressource `secret` dans votre conteneur. L'application peut alors accéder à votre certificat. Avec ce mode, il n'y a aucun chiffrement entre le client et le routeur.

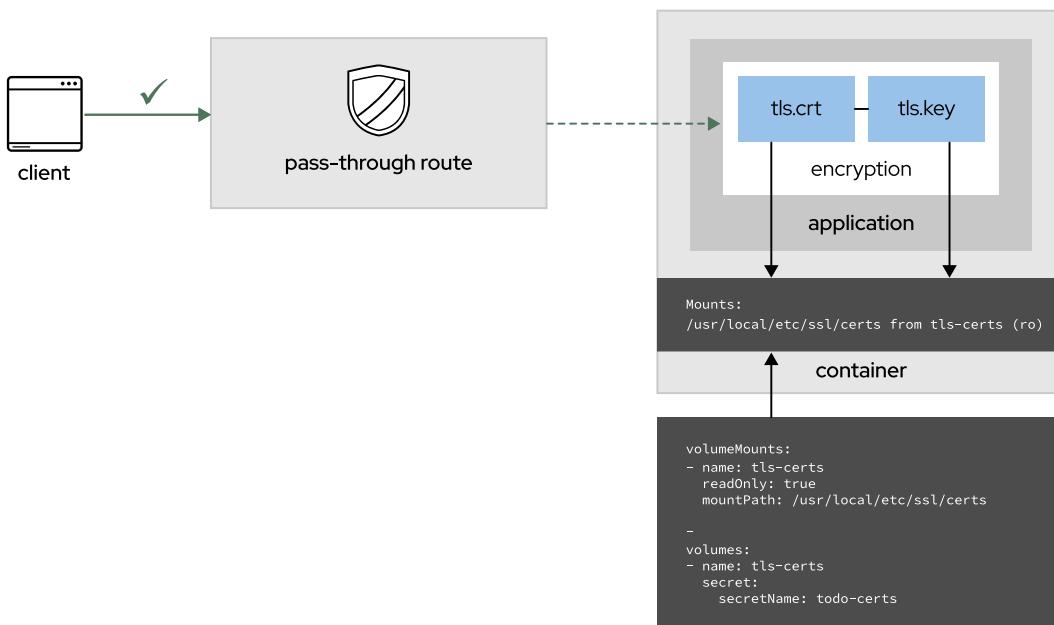


Figure 12.7: Sécurisation des applications avec des routes directes



## Références

Pour plus d'informations sur la manière de gérer les routes, reportez-vous au chapitre *Configuring Routes* de la documentation *Networking* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/networking/index#configuring-routes](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/networking/index#configuring-routes)

Pour plus d'informations sur la manière de configurer le trafic du cluster d'entrée, reportez-vous au chapitre *Configuring ingress cluster traffic* de la documentation *Networking* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/networking/index#configuring-ingress-cluster-traffic](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/networking/index#configuring-ingress-cluster-traffic)

## ► Exercice guidé

# Rendre les applications accessibles en externe

Dans cet exercice, vous allez exposer une application sécurisée par des certificats TLS.

### Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Déployer une application et lui créer un itinéraire non chiffré.
- Créer un itinéraire de périphérie OpenShift avec un chiffrement.
- Mettre à jour un déploiement OpenShift pour prendre en charge une nouvelle version de l'application.
- Créer un secret TLS OpenShift et le monter sur votre application.
- Vérifier que la communication avec l'application est chiffrée.

### Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée le projet OpenShift `network-ingress`. Elle permet également à l'utilisateur `developer` de modifier l'accès au projet.

```
[student@workstation ~]$ lab network-ingress start
```

### Instructions

En tant que développeur d'applications, vous êtes prêt à déployer votre application dans OpenShift. Dans cette activité, vous allez déployer deux versions de l'application, l'une exposée sur un trafic non chiffré (HTTP) et l'autre exposée sur un trafic sécurisé.

L'image de conteneur, accessible à l'adresse `https://quay.io/redhat/training/todo-angular`, possède deux balises : `v1.1`, qui est la version non sécurisée de l'application et `v1.2`, qui est la version sécurisée. Votre organisation utilise sa propre autorité de certification (CA) qui peut signer des certificats pour les domaines suivants :

- `*.apps.ocp4.example.com`
- `*.ocp4.example.com`

Le certificat de l'autorité de certification est accessible à l'adresse `~/D0280/labs/network-ingress/certs/training-CA.pem`. Le fichier `passphrase.txt` contient un mot de passe unique qui protège la clé de l'autorité de certification. Le dossier `certs` contient également la clé de l'autorité de certification.

► 1. Connectez-vous au cluster OpenShift et créez le projet `network-ingress`.

1.1. Connectez-vous au cluster en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
>   https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

1.2. Créez le projet `network-ingress`.

```
[student@workstation ~]$ oc new-project network-ingress
Now using project "network-ingress" on server "https://api.ocp4.example.com:6443".

...output omitted...
```

► 2. Le fichier de déploiement OpenShift de l'application est accessible à l'adresse `~/D0280/labs/network-ingress/todo-app-v1.yaml`. Le déploiement pointe vers `quay.io/redhattraining/todo-angular:v1.1`, qui est la version initiale et non chiffrée de l'application. Le fichier définit le service `todo-http` qui pointe vers le pod d'application.

Créez l'application et exposez le service.

2.1. Utilisez la commande `oc create` pour déployer l'application dans le projet OpenShift `network-ingress`.

```
[student@workstation ~]$ oc create -f \
>   ~/D0280/labs/network-ingress/todo-app-v1.yaml
deployment.apps/todo-http created
service/todo-http created
```

2.2. Attendez quelques minutes, afin que l'application puisse démarrer, puis examinez les ressources du projet.

```
[student@workstation ~]$ oc status
In project network-ingress on server https://api.ocp4.example.com:6443

svc/todo-http - 172.30.247.75:80 -> 8080
  deployment/todo-http deploys quay.io/redhattraining/todo-angular:v1.1
    deployment #1 running for 16 seconds - 1 pod
...output omitted...
```

2.3. Exécutez la commande `oc expose` pour créer une route permettant d'accéder à l'application. Attribuez à la route le nom d'hôte `todo-http.apps.ocp4.example.com`.

```
[student@workstation ~]$ oc expose svc todo-http \
>   --hostname todo-http.apps.ocp4.example.com
route.route.openshift.io/todo-http exposed
```

2.4. Récupérez le nom de la route et copiez-le dans le presse-papiers.

```
[student@workstation ~]$ oc get routes
NAME      HOST/PORT          PATH  SERVICES  PORT  ...
todo-http  todo-http.apps.ocp4.example.com        todo-http  8080  ...
```

- 2.5. Sur la machine `workstation`, ouvrez Firefox et accédez à l'URL de l'application. Vérifiez que vous pouvez voir l'application.
- `http://todo-http.apps.ocp4.example.com`
- 2.6. Ouvrez un nouvel onglet de terminal et exécutez la commande `tcpdump` avec les options suivantes pour intercepter le trafic sur le port 80 :
- `-i eth0` intercepte le trafic sur l'interface principale.
  - `-A` supprime les en-têtes et imprime les paquets au format ASCII.
  - `-n` désactive la résolution DNS.
  - `port 80` correspond au port de l'application.

Si vous le souhaitez, la commande `grep` vous permet de filtrer sur des ressources JavaScript.

Commencez par récupérer le nom de l'interface principale dont l'adresse IP est `172.25.250.9`.

```
[student@workstation ~]$ ip addr | grep 172.25.250.9
inet 172.25.250.9/24 brd 172.25.250.255 scope global noprefixroute eth0

[student@workstation ~]$ sudo tcpdump -i eth0 -A -n port 80 | grep "angular"
```



### Note

La commande complète est disponible à l'adresse `~/D0280/labs/network-ingress/tcpdump-command.txt`.

- 2.7. Sur Firefox, actualisez la page et observez l'activité dans le terminal. Appuyez sur `Ctrl+C` pour arrêter la capture.

```
...output omitted...
<script type="text/javascript" src="assets/js/libs/angular/angular.min.js">
<script type="text/javascript" src="assets/js/libs/angular/angular-route.min.js">
<script type="text/javascript" src="assets/js/libs/angular/angular-animate.min.js">
...output omitted...
```

- 3. Créez une route de périphérie sécurisée. Les certificats de périphérie chiffrent le trafic entre le client et le routeur, mais pas le trafic entre le routeur et le service. OpenShift génère son propre certificat qu'il signe avec son autorité de certification.

Dans les étapes suivantes, vous extrairez l'autorité de certification pour vous assurer que le certificat de route est signé.

- 3.1. Accédez à `~/D0280/labs/network-ingress` et exécutez la commande `oc create route` pour définir la nouvelle route.

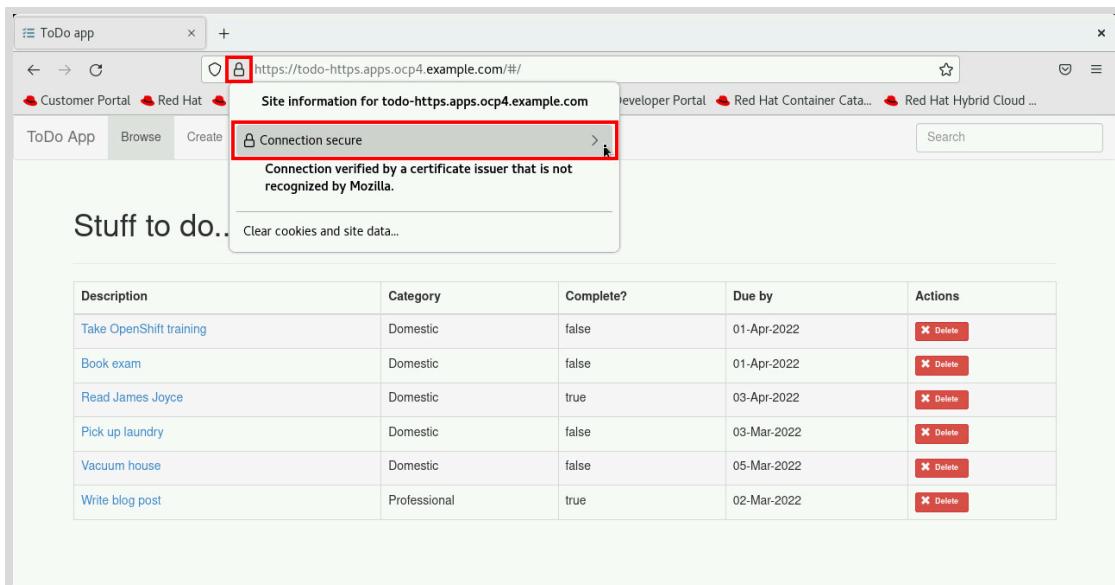
Attribuez à la route le nom d'hôte `todo-https.apps.ocp4.example.com`.

```
[student@workstation ~]$ cd ~/DO280/labs/network-ingress
[student@workstation network-ingress]$ oc create route edge todo-https \
>   --service todo-http \
>   --hostname todo-https.apps.ocp4.example.com
route.route.openshift.io/todo-https created
```

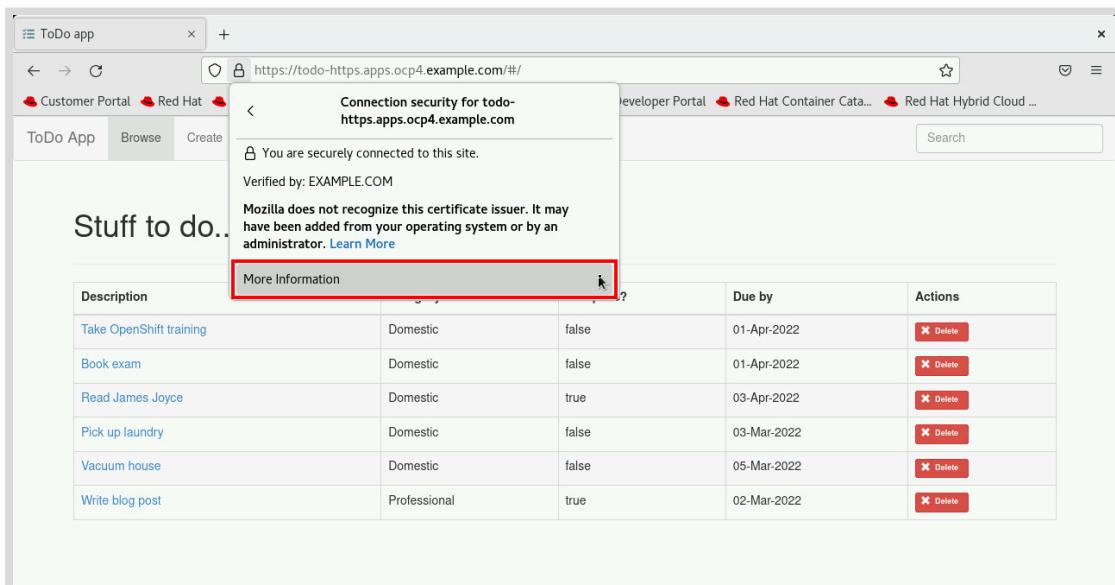
- 3.2. Pour tester la route et lire le certificat, ouvrez Firefox et accédez à l'URL de l'application.

- <https://todo-https.apps.ocp4.example.com>

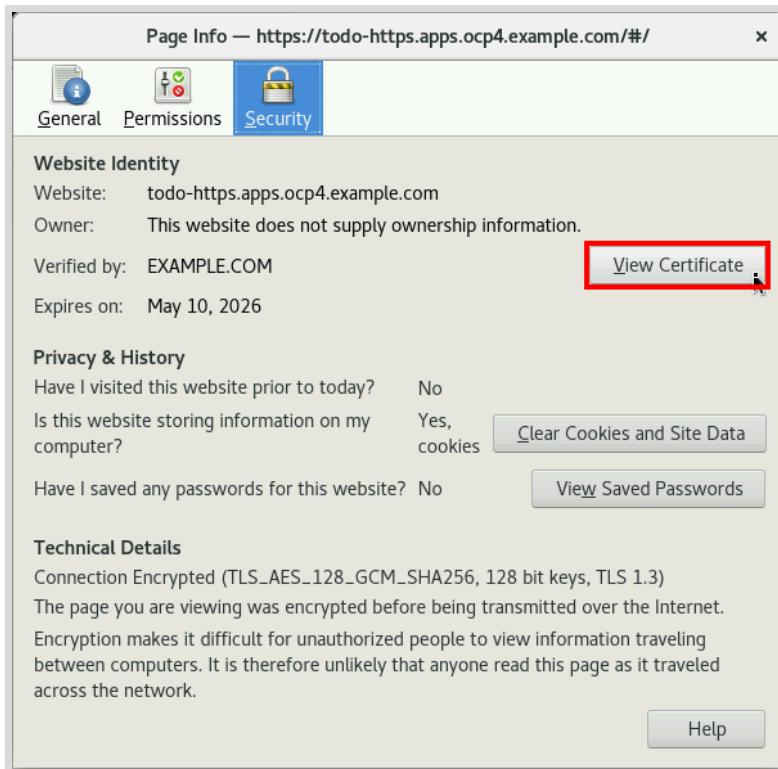
Cliquez sur le padlock, puis sur la flèche en regard de Connection secure.



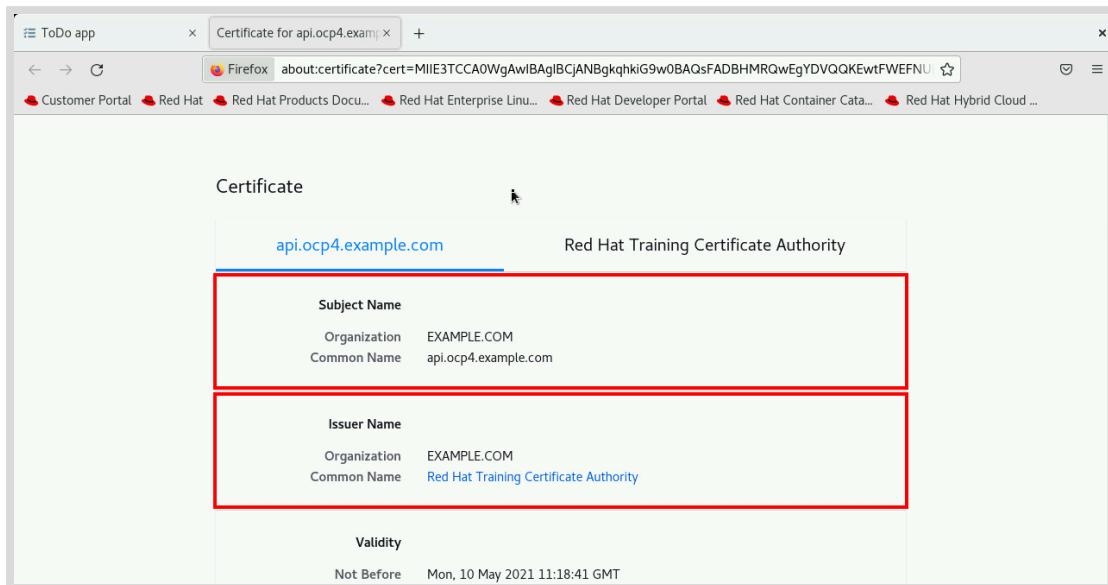
Cliquez sur More Information pour afficher la fenêtre d'informations de la page.



Cliquez sur View Certificate pour afficher les informations du certificat.



Localisez l'entrée CN pour constater que l'opérateur d'entrée OpenShift a créé le certificat avec sa propre autorité de certification.



- 3.3. À partir du terminal, utilisez la commande `curl` avec les options `-I` et `-v` pour récupérer les en-têtes de connexion.

La section `Server certificate` affiche des informations sur le certificat et l'autre nom correspond au nom de la route. La sortie indique que le certificat distant est approuvé, car il correspond à l'autorité de certification.

```
[student@workstation network-ingress]$ curl -I -v \
> https://todo-https.apps.ocp4.example.com
...output omitted...
* Server certificate:
*   subject: O=EXAMPLE.COM; CN=.api.ocp4.example.com
*   start date: May 10 11:18:41 2021 GMT
*   expire date: May 10 11:18:41 2026 GMT
*   subjectAltName: host « todo-https.apps.ocp4.example.com » matched cert's
  "*.apps.ocp4.example.com"
*   issuer: O=EXAMPLE.COM; CN=Red Hat Training Certificate Authority
*   SSL certificate verify ok.
...output omitted...
```

- 3.4. Bien que le trafic soit chiffré à la périphérie avec un certificat, vous pouvez toujours accéder au trafic non sécurisé au niveau du service, car le pod qui sous-tend le service ne propose pas de route chiffrée.

Récupérez l'adresse IP du service todo-http.

```
[student@workstation network-ingress]$ oc get svc todo-http \
> -o jsonpath="{.spec.clusterIP}{'\n'}"
172.30.102.29
```

- 3.5. Créez un pod de débogage dans le déploiement todo-http. Utilisez l'image de base universelle (UBI) de Red Hat, qui contient des outils de base pour interagir avec les conteneurs.

```
[student@workstation network-ingress]$ oc debug -t deployment/todo-http \
> --image registry.access.redhat.com/ubi8/ubi:8.4
Starting pod/todo-http-debug ...
Pod IP: 10.131.0.255
If you don't see a command prompt, try pressing enter.
sh-4.4$
```

- 3.6. À partir du pod de débogage, utilisez la commande `curl` pour accéder au service via HTTP. Remplacez l'adresse IP par celle obtenue lors d'une étape précédente.

La sortie indique que l'application est disponible via HTTP.

```
sh-4.4$ curl -v 172.30.102.29
* Rebuilt URL to: 172.30.102.29/
*   Trying 172.30.102.29...
* TCP_NODELAY set
* Connected to 172.30.102.29 (172.30.102.29) port 80 (#0)
> GET / HTTP/1.1
> Host: 172.30.102.29
> User-Agent: curl/7.61.1
> Accept: /
>
< HTTP/1.1 200 OK
...output omitted...
```

- 3.7. Quittez le pod de débogage.

```
sh-4.4$ exit  
Removing debug pod ...
```

- 3.8. Supprimez la route de périphérie. Dans les étapes suivantes, vous allez définir la route directe.

```
[student@workstation network-ingress]$ oc delete route todo-https  
route.route.openshift.io "todo-https" deleted
```

► 4. Générez des certificats TLS pour l'application.

Dans les étapes suivantes, vous allez générer un certificat signé par l'autorité de certification que vous joignez comme secret au pod. Vous configurerez ensuite une route sécurisée en mode direct et laisserez l'application exposer ce certificat.

- 4.1. Accédez au répertoire ~/D0280/labs/network-ingress/certs et listez les fichiers.

```
[student@workstation network-ingress]$ cd certs  
[student@workstation certs]$ ls -l  
total 20  
-rw-rw-r-- 1 student student 604 Nov 29 17:35 openssl-commands.txt  
-rw-r--r-- 1 student student 33 Nov 29 17:35 passphrase.txt  
-rw-r--r-- 1 student student 1743 Nov 29 17:35 training-CA.key  
-rw-r--r-- 1 student student 1363 Nov 29 17:35 training-CA.pem  
-rw-r--r-- 1 student student 406 Nov 29 17:35 training.ext
```

- 4.2. Générez la clé privée pour votre certificat signé par l'autorité de certification.



**Note**

Les commandes suivantes permettant de générer un certificat signé sont toutes disponibles dans le fichier ~/D0280/labs/network-ingress/certs/openssl-commands.txt.

```
[student@workstation certs]$ openssl genrsa -out training.key 4096  
Generating RSA private key, 4096 bit long modulus (2 primes)  
...output omitted...  
e is 65537 (0x010001)
```

- 4.3. Générez la demande de signature de certificat (CSR) pour todo-https.apps.ocp4.example.com.

```
[student@workstation certs]$ openssl req -new \  
> -key training.key -out training.csr \  
> -subj "/C=US/ST=North Carolina/L=Raleigh/O=Red Hat/\\  
> CN=todo-https.apps.ocp4.example.com"
```

**Mise en garde**

Make sure to type the **subject** of the request on one line. Sinon, supprimez l'option **-subj** et son contenu. En l'absence de l'option **-subj**, la commande **openssl** vous invite à saisir des valeurs ; veillez à indiquer le nom courant (CN) **todo-https.apps.ocp4.example.com**.

- 4.4. Pour terminer, générez le certificat signé. Notez l'utilisation des options **-CA** et **-CAkey** pour signer le certificat par rapport à l'autorité de certification. L'option **-passin** vous permet de réutiliser le mot de passe de l'autorité de certification. L'option **extfile** vous permet de définir un *autre nom de l'objet* (SAN).

```
[student@workstation certs]$ openssl x509 -req -in training.csr \
> -passin file:passphrase.txt \
> -CA training-CA.pem -CAkey training-CA.key -CAcreateserial \
> -out training.crt -days 1825 -sha256 -extfile training.ext
Signature ok
subject=C = US, ST = North Carolina, L = Raleigh, O = Red Hat, CN = todo-
https.apps.ocp4.example.com
Getting CA Private Key
```

- 4.5. Assurez-vous que le certificat et la clé que vous venez de créer se trouvent dans le répertoire actif.

```
[student@workstation certs]$ ls -lrt
total 36
-rw-r--r-- 1 student student 599 Jul 31 09:35 openssl-commands.txt
-rw-r--r-- 1 student student 33 Aug 3 12:38 passphrase.txt
-rw-r--r-- 1 student student 352 Aug 3 12:38 training.ext
-rw----- 1 student student 1743 Aug 3 12:38 training-CA.key
-rw-r--r-- 1 student student 1334 Aug 3 12:38 training-CA.pem
-rw----- 1 student student 1675 Aug 3 13:38 training.key
-rw-rw-r-- 1 student student 1017 Aug 3 13:39 training.csr
-rw-rw-r-- 1 student student 41 Aug 3 13:40 training-CA.srl
-rw-rw-r-- 1 student student 1399 Aug 3 13:40 training.crt
```

- 4.6. Revenez au répertoire **network-ingress**. Cette étape est importante, car la suivante implique la création d'un itinéraire à l'aide du certificat autosigné.

```
[student@workstation certs]$ cd ~/DO280/labs/network-ingress
```

- 5. Déployez une nouvelle version de votre application.

La nouvelle version de l'application requiert un certificat et une clé à l'intérieur du conteneur, qui se trouve à l'adresse **/usr/local/etc/ssl/certs**. Le serveur Web de cette version est configuré avec une prise en charge de SSL. Créez un secret pour importer le certificat à partir de la machine **workstation**. Dans une étape ultérieure, le déploiement d'applications demande ce secret et expose son contenu au conteneur qui se trouve à l'adresse **/usr/local/etc/ssl/certs**.

**chapitre 12 |** Configuration de la mise en réseau OpenShift pour les applications

- 5.1. Créez un secret OpenShift `tls` nommé `todo-certs`. Utilisez les options `--cert` et `--key` pour intégrer les certificats TLS. Utilisez le fichier `training.crt` comme certificat et `training.key` comme clé.

```
[student@workstation network-ingress]$ oc create secret tls todo-certs \
>   --cert certs/training.crt --key certs/training.key
secret/todo-certs created
```

- 5.2. Le fichier de déploiement, accessible à l'adresse `~/DO280/labs/network-ingress/todo-app-v2.yaml`, pointe vers la version 2 de l'image de conteneur. La nouvelle version de l'application est configurée pour prendre en charge les certificats SSL. Exécutez `oc create` pour créer un nouveau déploiement à l'aide de cette image.

```
[student@workstation network-ingress]$ oc create -f todo-app-v2.yaml
deployment.apps/todo-https created
service/todo-https created
```

- 5.3. Patientez quelques minutes pour vous assurer que le pod d'application s'exécute. Copiez le nom du pod dans votre presse-papiers.

```
[student@workstation network-ingress]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
...output omitted...
todo-https-59d8fc9d47-265ds   1/1     Running   0          62s
```

- 5.4. Examinez les volumes qui sont montés à l'intérieur du pod. La sortie indique que les certificats sont montés dans `/usr/local/etc/ssl/certs`.

```
[student@workstation network-ingress]$ oc describe pod \
> todo-https-59d8fc9d47-265ds | grep -A2 Mounts
Mounts:
  /usr/local/etc/ssl/certs from tls-certs (ro)
  /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-qrtkj (ro)
```

**▶ 6.** Créez la route sécurisée.

- 6.1. Exécutez la commande `oc create route` pour définir la nouvelle route. Attribuez à la route le nom d'hôte `todo-https.apps.ocp4.example.com`.

```
[student@workstation network-ingress]$ oc create route passthrough todo-https \
>   --service todo-https --port 8443 \
>   --hostname todo-https.apps.ocp4.example.com
route.route.openshift.io/todo-https created
```

- 6.2. Utilisez la commande `curl` en mode détaillé pour tester la route et lire le certificat. Utilisez l'option `--cacert` pour transmettre le certificat d'autorité de certification à la commande `curl`.

La sortie indique une correspondance entre la chaîne de certificats et le certificat de l'application. Cette correspondance indique que le routeur OpenShift ne transfère que les paquets chiffrés par le certificat du serveur Web de l'application.

```
[student@workstation network-ingress]$ curl -vv -I \
>   --cacert certs/training-CA.pem \
>   https://todo-https.apps.ocp4.example.com
...output omitted...
* Server certificate:
*   subject: C=US; ST=North Carolina; L=Raleigh; O=Red Hat; CN=todo-
https.apps.ocp4.example.com
*   start date: Jun 15 01:53:30 2021 GMT
*   expire date: Jun 14 01:53:30 2026 GMT
*   subjectAltName: host "todo-https.apps.ocp4.example.com" matched cert's
"*.apps.ocp4.example.com"
*   issuer: C=US; ST=North Carolina; L=Raleigh; O=Red Hat; CN=ocp4.example.com
*   SSL certificate verify ok.
...output omitted...
```

- ▶ 7. Créez un nouveau pod de débogage pour confirmer davantage le bon chiffrement au niveau du service.

7.1. Récupérez l'adresse IP du service todo-https.

```
[student@workstation network-ingress]$ oc get svc todo-https \
>   -o jsonpath="{.spec.clusterIP}{'\n'}"
172.30.121.154
```

7.2. Créez un pod de débogage dans le déploiement todo-https avec l'UBI de Red Hat.

```
[student@workstation network-ingress]$ oc debug -t deployment/todo-https \
>   --image registry.access.redhat.com/ubi8/ubi:8.4
Starting pod/todo-https-debug ...
Pod IP: 10.128.2.129
If you don't see a command prompt, try pressing enter.
sh-4.4$
```

- 7.3. À partir du pod de débogage, utilisez la commande `curl` pour accéder au service via HTTP. Remplacez l'adresse IP par celle obtenue lors d'une étape précédente.

La sortie indique que l'application n'est pas disponible via HTTP et que le serveur Web vous redirige vers la version sécurisée.

```
sh-4.4$ curl -I http://172.30.121.154
HTTP/1.1 301 Moved Permanently
Server: nginx/1.14.1
Date: Tue, 15 Jun 2021 02:01:19 GMT
Content-Type: text/html
Connection: keep-alive
Location: https://172.30.121.154:8443/
```

- 7.4. Enfin, accédez à l'application via HTTPS. Utilisez l'option `-k`, car le conteneur n'a pas accès au certificat de l'autorité de certification.

```
sh-4.4$ curl -s -k https://172.30.121.154:8443 | head -n5
<!DOCTYPE html>
<html lang="en" ng-app="todoItemsApp" ng-controller="appCtl">
<head>
    <meta charset="utf-8">
    <title>ToDo app</title>
```

7.5. Quittez le pod de débogage.

```
sh-4.4$ exit
Removing debug pod ...
```

► 8. Effectuez un nettoyage.

8.1. Accédez au répertoire personnel.

```
[student@workstation network-ingress]$ cd
```

8.2. Supprimez le projet network-ingress.

```
[student@workstation ~]$ oc delete project network-ingress
project.project.openshift.io "network-ingress" deleted
```

## Fin

Sur la machine **workstation**, utilisez la commande **lab** pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab network-ingress finish
```

L'exercice guidé est maintenant terminé.

# Configuration des politiques réseau

## Résultats

Après avoir terminé cette section, vous devez pouvoir limiter le trafic réseau entre les projets et les pods.

## Gestion des politiques réseau dans OpenShift

Les politiques réseau vous permettent de configurer des politiques d'isolement pour chacun des pods. Les politiques réseau n'ont pas besoin de priviléges d'administration, ce qui confère aux développeurs un meilleur contrôle des applications dans leurs projets. Vous pouvez utiliser des politiques réseau pour créer, dans le SDN, des zones logiques correspondant aux zones réseau de votre entreprise. L'avantage de cette méthode réside dans le fait que l'emplacement des pods en cours d'exécution n'a plus d'importance, étant donné que les politiques réseau vous permettent de séparer le trafic, d'où qu'il provienne.

Pour gérer la communication réseau entre deux espaces de noms, affectez un libellé à l'espace de noms qui doit accéder à un autre. La commande suivante affecte le libellé `name=network-1` à l'espace de noms `network-1`:

```
[user@host ~]$ oc label namespace network-1 name=network-1
```

Les exemples suivants décrivent les politiques réseau qui permettent une communication entre les espaces de noms `network-1` et `network-2`:

- La politique réseau suivante s'applique à tous les pods ayant le libellé `deployment="product-catalog"` dans l'espace de noms `network-1`. La politique autorise le trafic TCP sur le port 8080 à partir des pods ayant le libellé `role="qa"` dans l'espace de noms `network-2`.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: network-1-policy
spec:
  podSelector: ①
    matchLabels:
      deployment: product-catalog
  ingress: ②
    - from: ③
      - namespaceSelector:
          matchLabels:
            name: network-2
  podSelector:
    matchLabels:
      role: qa
```

```
ports: ④
  - port: 8080
    protocol: TCP
```

- ❶ Le champ `podSelector` de niveau supérieur est requis et définit les pods qui utilisent la politique réseau. Si le champ `podSelector` est vide, tous les pods de l'espace de noms sont mis en correspondance.
- ❷ Le champ `ingress` définit une liste de règles de trafic entrant à appliquer aux pods correspondants à partir du champ `podSelector` de niveau supérieur.
- ❸ Le champ `from` définit une liste de règles pour faire correspondre le trafic provenant de toutes les sources. Les sélecteurs ne se limitent pas au projet dans lequel la politique réseau est définie.
- ❹ Le champ `ports` contient la liste des ports de destination qui autorisent le trafic à accéder aux pods sélectionnés.
- La politique réseau suivante autorise le trafic en provenance de tous les pods et ports de l'espace de noms `network-1` vers tous les pods et ports de l'espace de noms `network-2`. Cette politique est moins restrictive que `network-1`, dans la mesure où elle ne limite pas le trafic en provenance des pods de l'espace de noms `network-1`.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: network-2-policy
spec:
  podSelector: {}
  ingress:
    - from:
        - namespaceSelector:
          matchLabels:
            name: network-1
```

**Note**

Les politiques réseau sont des ressources Kubernetes. Cela signifie donc que vous pouvez les gérer à l'aide de commandes oc.

L'un des avantages liés à l'utilisation des politiques réseau est la gestion de la sécurité entre des projets (tenants), ce qui s'avère impossible avec les technologies de couche 2 telles que les réseaux VLAN. Cette méthode vous permet de créer des politiques personnalisées entre des projets pour vous assurer que les utilisateurs ne peuvent accéder qu'au contenu qui leur est destiné (ce qui est conforme au principe de moindre privilège).

Les champs de la politique réseau qui utilisent une liste d'objets peuvent être combinés dans le même objet ou listés en tant qu'objets multiples. S'ils sont combinés, les conditions sont combinées à l'aide d'un ANDlogique. S'ils sont séparés dans une liste, les conditions sont combinées à l'aide d'un ORlogique. Les options de logique vous permettent de créer des règles de politique très spécifiques. Les exemples suivants illustrent les différences que la syntaxe permet de réaliser :

- Cet exemple regroupe les sélecteurs dans une seule règle, ce qui autorise uniquement l'accès à partir des pods de l'espace de noms `dev` avec le libellé `app=mobile`. Il s'agit d'un exemple d'opérateur `AND` logique.

```
...output omitted...
ingress:
- from:
  - namespaceSelector:
    matchLabels:
      name: dev
  podSelector:
    matchLabels:
      app: mobile
```

- En modifiant le champ `podSelector` dans l'exemple précédent pour qu'il s'agisse d'un élément de la liste `from`, tous les pods de l'espace de noms `dev` et tous ceux des espaces de noms ayant le libellé `app=mobile` peuvent accéder aux pods correspondant au champ `podSelector` de niveau supérieur. Il s'agit d'un exemple d'opérateur `OR` logique.

```
...output omitted...
ingress:
- from:
  - namespaceSelector:
    matchLabels:
      name: dev
  - podSelector:
    matchLabels:
      app: mobile
```

Si un pod correspond à des sélecteurs dans une ou plusieurs politiques réseau, il n'acceptera que les connexions qui sont autorisées par au moins l'une de ces politiques réseau. Un exemple strict est une politique qui consiste à refuser tout le trafic entrant vers les pods de votre projet, y compris à partir d'autres pods à l'intérieur de votre projet. Un sélecteur de pod vide signifie que cette politique s'applique à tous les pods de ce projet. La politique suivante bloque tout le trafic, car aucune règle d'entrée n'est définie. Le trafic est bloqué, sauf si vous définissez une politique explicite qui ignore ce comportement par défaut.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: default-deny
spec:
  podSelector: {}
```

Si vous disposez d'une surveillance de cluster ou de routes exposées, vous devez également autoriser l'entrée à partir de ces éléments. Les politiques suivantes autorisent l'entrée à partir des contrôleurs d'entrée et de surveillance OpenShift :

```
---
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: allow-from-openshift-ingress
```

```
spec:  
  podSelector: {}  
  ingress:  
    - from:  
      - namespaceSelector:  
          matchLabels:  
            network.openshift.io/policy-group: ingress  
---  
apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy  
metadata:  
  name: allow-from-openshift-monitoring  
spec:  
  podSelector: {}  
  ingress:  
    - from:  
      - namespaceSelector:  
          matchLabels:  
            network.openshift.io/policy-group: monitoring
```



### Important

Si le contrôleur d'entrée `default` utilise la stratégie de publication des points d'accès `HostNetwork`, l'espace de noms `default` doit avoir le libellé `network.openshift.io/policy-group=ingress`.

Vérifiez la stratégie de publication des points d'accès à l'aide de la commande `oc describe` pour décrire la ressource `ingresscontroller/default` dans l'espace de noms `openshift-ingress-operator`.



### Références

Pour plus d'informations sur la politique réseau, reportez-vous au chapitre *Network policy* de la documentation *Networking* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/networking/index#network-policy](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/networking/index#network-policy)

## ► Exercice guidé

# Configuration des politiques réseau

Dans cet exercice, vous allez créer des politiques réseau et examiner l'isolement des pods créé par ces politiques.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Créer des politiques réseau pour contrôler la communication entre les pods.
- Vérifier que le trafic entrant est limité aux pods.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande s'assure que l'environnement est prêt et télécharge les fichiers de ressources nécessaires à cet exercice.

```
[student@workstation ~]$ lab network-policy start
```

## Instructions

- 1. Connectez-vous au cluster OpenShift et créez le projet `network-policy`.

- 1.1. Connectez-vous au cluster en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Créez le projet `network-policy`.

```
[student@workstation ~]$ oc new-project network-policy
Now using project "network-policy" on server "https://api.ocp4.example.com:6443".

...output omitted...
```

- 2. Créez deux déploiements et créez une route pour l'un d'eux.

- 2.1. Créez le déploiement `hello` à l'aide de l'image de conteneur `quay.io/redhat-training/hello-world-nginx:v1.0`.

```
[student@workstation ~]$ oc new-app --name hello \
>   --image quay.io/redhattraining/hello-world-nginx:v1.0
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "hello" created
  deployment.apps "hello" created
  service "hello" created
--> Success
...output omitted...
```

- 2.2. Créez le déploiement **test** à l'aide de l'image de conteneur `quay.io/redhattraining/hello-world-nginx:v1.0`.

```
[student@workstation ~]$ oc new-app --name test \
>   --image quay.io/redhattraining/hello-world-nginx:v1.0
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "test" created
  deployment.apps "test" created
  service "test" created
--> Success
...output omitted...
```

- 2.3. Utilisez la commande `oc expose` pour créer une route vers le service **hello**.

```
[student@workstation ~]$ oc expose service hello
route.route.openshift.io/hello exposed
```

► 3. Vérifiez l'accès au pod **hello** avec les commandes `oc rsh` et `curl`.

- 3.1. Ouvrez un deuxième terminal et exécutez le script situé dans `~/DO280/labs/network-policy/display-project-info.sh`. Ce script fournit des informations sur les pods, le service et la route utilisés dans le reste de cet exercice.

```
[student@workstation ~]$ ~/DO280/labs/network-policy/display-project-info.sh
=====
PROJECT: network-policy

POD NAME          IP ADDRESS
hello-6c4984d949-g28c4  10.8.0.13
test-c4d74c9d5-5pq9s  10.8.0.14

SERVICE NAME    CLUSTER-IP
hello           172.30.137.226
test            172.30.159.119

ROUTE NAME      HOSTNAME                      PORT
hello           hello-network-policy.apps.ocp4.example.com  8080-tcp
=====
```

- 3.2. Utilisez les commandes `oc rsh` et `curl` pour vérifier que le pod **test** peut accéder à l'adresse IP du pod **hello**.

```
[student@workstation ~]$ oc rsh test-c4d74c9d5-5pq9s \
>   curl 10.8.0.13:8080 | grep Hello
<h1>Hello, world from nginx!</h1>
```

- 3.3. Utilisez les commandes `oc rsh` et `curl` pour vérifier que le pod `test` peut accéder à l'adresse IP du service `hello`.

```
[student@workstation ~]$ oc rsh test-c4d74c9d5-5pq9s \
>   curl 172.30.137.226:8080 | grep Hello
<h1>Hello, world from nginx!</h1>
```

- 3.4. Vérifiez l'accès au pod `hello` à l'aide de la commande `curl` sur l'URL de la route `hello`.

```
[student@workstation ~]$ curl -s hello-network-policy.apps.ocp4.example.com | \
>   grep Hello
<h1>Hello, world from nginx!</h1>
```

► 4. Créez le projet `network-test` et un déploiement nommé `sample-app`.

- 4.1. Créez le projet `network-test`.

```
[student@workstation ~]$ oc new-project network-test
Now using project "network-test" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 4.2. Créez le déploiement `sample-app` avec l'image `quay.io/redhattraining/hello-world-nginx:v1.0`. L'application Web écoute sur le port 8080.

```
[student@workstation ~]$ oc new-app --name sample-app \
>   --image quay.io/redhattraining/hello-world-nginx:v1.0
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "sample-app" created
  deployment.apps "sample-app" created
  service "sample-app" created
--> Success
...output omitted...
```

► 5. Vérifiez que les pods d'un espace de noms différent peuvent accéder aux pods `hello` et `test` de l'espace de noms `network-policy`.

- 5.1. Dans le deuxième terminal, exécutez à nouveau le script `display-project-info.sh` pour afficher le nom complet du pod `sample-app`.

```
[student@workstation ~]$ ~/D0280/labs/network-policy/display-project-info.sh
=====
PROJECT: network-policy

POD NAME          IP ADDRESS
hello-6c4984d949-g28c4  10.8.0.13
```

```
test-c4d74c9d5-5pq9s      10.8.0.14

SERVICE NAME   CLUSTER-IP
hello          172.30.137.226
test           172.30.159.119

ROUTE NAME     HOSTNAME                  PORT
hello          hello-network-policy.apps.ocp4.example.com  8080-tcp
=====
PROJECT: network-test

POD NAME
sample-app-d5f945-spx9q
=====
```

- 5.2. De retour dans le premier terminal, utilisez les commandes `oc rsh` et `curl` pour vérifier que le pod `sample-app` peut accéder à l'adresse IP du pod `hello`.

```
[student@workstation ~]$ oc rsh sample-app-d5f945-spx9q \
>   curl 10.8.0.13:8080 | grep Hello
<h1>Hello, world from nginx!</h1>
```

- 5.3. Utilisez les commandes `oc rsh` et `curl` pour vérifier l'accès au pod `test` à partir du pod `sample-app`. Ciblez l'adresse IP récupérée précédemment pour le pod `test`.

```
[student@workstation ~]$ oc rsh sample-app-d5f945-spx9q \
>   curl 10.8.0.14:8080 | grep Hello
<h1>Hello, world from nginx!</h1>
```

- 6. À partir du projet `network-policy`, créez la politique réseau `deny-all` à l'aide du fichier de ressources disponible à l'emplacement suivant : `~/D0280/labs/network-policy/deny-all.yaml`.

- 6.1. Basculez vers le projet `network-policy`.

```
[student@workstation ~]$ oc project network-policy
Now using project "network-policy" on server "https://api.ocp4.example.com:6443".
```

- 6.2. Accédez au répertoire `~/D0280/labs/network-policy`.

```
[student@workstation ~]$ cd ~/D0280/labs/network-policy
```

- 6.3. Utilisez un éditeur de texte pour mettre à jour le fichier `deny-all.yaml` avec un champ `podSelector` vide afin de cibler tous les pods de l'espace de noms.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-all
spec:
  podSelector: {}
```

**Note**

Une solution est fournie à l'emplacement suivant : ~/D0280/solutions/network-policy/deny-all.yaml.

- 6.4. Créez la politique réseau à l'aide de la commande `oc create`.

```
[student@workstation network-policy]$ oc create -f deny-all.yaml
networkpolicy.networking.k8s.io/deny-all created
```

- ▶ 7. Vérifiez qu'il n'y a plus d'accès aux pods dans l'espace de noms `network-policy`.

- 7.1. Vérifiez qu'il n'y a plus d'accès au pod `hello` via la route exposée. Attendez quelques secondes, puis appuyez sur `Ctrl+C` pour quitter la commande `curl` qui ne répond pas.

```
[student@workstation network-policy]$ curl -s \
>   hello-network-policy.apps.ocp4.example.com | grep Hello
^C
```

**Important**

Si le pod `hello` arrive sur le même nœud qu'un pod `router-default`, la commande `curl` fonctionne lorsque le trafic transite par ce pod de routeur. C'est uniquement le cas avec les clusters à trois nœuds. Dans un cluster OpenShift traditionnel, dans lequel les nœuds d'infrastructure ou de plan de contrôle sont séparés des nœuds de calcul, la politique réseau s'applique à tous les pods de routeur du cluster.

Si la commande `curl` réussit, exécutez-la à nouveau pour confirmer que la politique réseau fonctionne comme prévu. Cette seconde tentative doit transiter par l'autre pod de routeur dans le cluster.

- 7.2. Vérifiez que le pod `test` ne peut plus accéder à l'adresse IP du pod `hello`. Attendez quelques secondes, puis appuyez sur `Ctrl+C` pour quitter la commande `curl` qui ne répond pas.

```
[student@workstation network-policy]$ oc rsh test-c4d74c9d5-5pq9s \
>   curl 10.8.0.13:8080 | grep Hello
^C
command terminated with exit code 130
```

- 7.3. Basculez vers le projet `network-test`.

```
[student@workstation network-policy]$ oc project network-test
Now using project "network-test" on server "https://api.ocp4.example.com:6443".
```

- 7.4. Confirmez que le pod `sample-app` ne peut plus accéder à l'adresse IP du pod `test`. Attendez quelques secondes, puis appuyez sur `Ctrl+C` pour quitter la commande `curl` qui ne répond pas.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
>   curl 10.8.0.14:8080 | grep Hello
^C
command terminated with exit code 130
```

- 8. Créez une politique réseau pour autoriser le trafic vers le pod hello dans l'espace de noms `network-policy` à partir du pod `sample-app` dans l'espace de noms `network-test` sur TCP sur le port 8080. Utilisez le fichier de ressources disponible à l'emplacement suivant : `~/D0280/labs/network-policy/allow-specific.yaml`.
- 8.1. Utilisez un éditeur de texte pour remplacer les sections `CHANGE_ME` du fichier `allow-specific.yaml` comme suit.

```
...output omitted...
spec:
  podSelector:
    matchLabels:
      deployment: hello
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            name: network-test
        podSelector:
          matchLabels:
            deployment: sample-app
    ports:
      - port: 8080
        protocol: TCP
```

**Note**

Une solution est fournie à l'emplacement suivant : `~/D0280/solutions/network-policy/allow-specific.yaml`.

- 8.2. Créez la politique réseau à l'aide de la commande `oc create`.

```
[student@workstation network-policy]$ oc create -n network-policy -f \
>   allow-specific.yaml
networkpolicy.networking.k8s.io/allow-specific created
```

- 8.3. Affichez les politiques réseau dans l'espace de noms `network-policy`.

```
[student@workstation network-policy]$ oc get networkpolicies -n network-policy
NAME          POD-SELECTOR      AGE
allow-specific deployment=hello  11s
deny-all       <none>           5m6s
```

- 9. En tant qu'utilisateur `admin`, attribuez le libellé `network-test` à l'espace de noms `name=network-test`.

9.1. Connectez-vous en tant qu'utilisateur admin.

```
[student@workstation network-policy]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

9.2. Utilisez la commande `oc label` pour appliquer le libellé `name=network-test`.

```
[student@workstation network-policy]$ oc label namespace network-test \
>   name=network-test
namespace/network-test labeled
```



### Important

La politique réseau `allow-specific` utilise des libellés pour faire correspondre le nom d'un espace de noms. Par défaut, aucun libellé n'est attribué automatiquement aux espaces de noms ni aux projets.

9.3. Confirmez que le libellé a été appliqué.

```
[student@workstation network-policy]$ oc describe namespace network-test
Name:           network-test
Labels:         name=network-test
...output omitted...
```

9.4. Connectez-vous en tant qu'utilisateur developer

```
[student@workstation network-policy]$ oc login -u developer -p developer
Login successful.

...output omitted...
```

► 10. Vérifiez que le pod `sample-app` peut accéder à l'adresse IP du pod `hello`, mais pas à l'adresse IP du pod `test`.

10.1. Basculez vers le projet `network-test`.

```
[student@workstation network-policy]$ oc project network-test
Already on project "network-test" on server "https://api.ocp4.example.com:6443".
```

10.2. Vérifiez l'accès au pod `hello` dans l'espace de noms `network-policy`.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
>   curl 10.8.0.13:8080 | grep Hello
<h1>Hello, world from nginx!</h1>
```

10.3. Vérifiez l'absence de réponse du pod `hello` sur un autre port. Étant donné que la politique réseau autorise uniquement l'accès au port 8080 sur le pod `hello`, les demandes adressées à tout autre port sont ignorées et finissent par expirer. Attendez

**chapitre 12 |** Configuration de la mise en réseau OpenShift pour les applications

quelques secondes, puis appuyez sur Ctrl+C pour quitter la commande curl qui ne répond pas.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
>   curl 10.8.0.13:8181 | grep Hello
^C
command terminated with exit code 130
```

10.4. Vérifiez qu'il n'y a aucun accès au pod test. Attendez quelques secondes, puis appuyez sur Ctrl+C pour quitter la commande curl qui ne répond pas.

```
[student@workstation network-policy]$ oc rsh sample-app-d5f945-spx9q \
>   curl 10.8.0.14:8080 | grep Hello
^C
command terminated with exit code 130
```

- 11. Créez une politique réseau pour autoriser le trafic vers le pod hello à partir de la route exposée. Utilisez le fichier de ressources disponible à l'emplacement suivant : ~/D0280/labs/network-policy/allow-from-openshift-ingress.yaml.

- 11.1. Utilisez un éditeur de texte pour remplacer les valeurs CHANGE\_ME dans le fichier allow-from-openshift-ingress.yaml comme suit.

```
...output omitted...
spec:
  podSelector: {}
  ingress:
    - from:
        - namespaceSelector:
            matchLabels:
              network.openshift.io/policy-group: ingress
```

**Note**

Une solution est fournie à l'emplacement suivant : ~/D0280/solutions/network-policy/allow-from-openshift-ingress.yaml.

- 11.2. Créez la politique réseau à l'aide de la commande oc create.

```
[student@workstation network-policy]$ oc create -n network-policy -f \
>   allow-from-openshift-ingress.yaml
networkpolicy.networking.k8s.io/allow-from-openshift-ingress created
```

- 11.3. Affichez les politiques réseau dans l'espace de noms network-policy.

```
[student@workstation network-policy]$ oc get networkpolicies -n network-policy
NAME                  POD-SELECTOR      AGE
allow-from-openshift-ingress  <none>          10s
allow-specific          deployment=hello  8m16s
deny-all                <none>          13m
```

11.4. Connectez-vous en tant qu'utilisateur admin.

```
[student@workstation network-policy]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

11.5. Appliquez le libellé `network.openshift.io/policy-group=ingress` à l'espace de noms `default`.

```
[student@workstation network-policy]$ oc label namespace default \
>   network.openshift.io/policy-group=ingress
namespace/default labeled
```



### Note

L'application de ce libellé à l'espace de noms `default` n'est nécessaire que parce que le contrôleur d'entrée `default` de la salle de classe utilise la stratégie de publication des points d'accès `HostNetwork`.

11.6. Vérifiez l'accès au pod `hello` via la route exposée.

```
[student@workstation network-policy]$ curl -s \
>   hello-network-policy.apps.ocp4.example.com | grep Hello
<h1>Hello, world from nginx!</h1>
```

- ▶ 12. Fermez la fenêtre de terminal qui contient la sortie du script `display-project-info.sh`. Accédez au répertoire personnel.

```
[student@workstation network-policy]$ cd
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab network-policy finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Configuration de la mise en réseau OpenShift pour les applications

Dans cet atelier, vous allez configurer une route directe TLS pour votre application.

### Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Déployer une application et configurer une route non sécurisée.
- Limiter le trafic vers les applications.
- Générer un certificat TLS pour une application.
- Configurer une route directe pour une application avec un certificat TLS.

### Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée l'autorité de certification autosignée qui sera utilisée dans cet atelier.

```
[student@workstation ~]$ lab network-review start
```

### Instructions

Dans cette révision, vous déployez une application PHP qui imprime des informations sur le système. L'application est disponible selon deux configurations différentes : la première fonctionne avec un réseau non chiffré qui écoute sur le port 8080 et la deuxième utilise un certificat TLS pour chiffrer le trafic réseau, qui écoute sur le port 8443.

L'image de conteneur pour cette révision est accessible à l'adresse `quay.io/redhattraining/php-ssl`. Elle comporte deux balises : `v1.0` pour la version non sécurisée de l'application et `v1.1` pour la version sécurisée.

1. En tant qu'utilisateur `developer` d'OpenShift, créez le projet `network-review`.
2. En tant qu'utilisateur `developer`, déployez la version non sécurisée de l'application PHP dans le projet `network-review`. Utilisez le fichier de ressources disponible à l'emplacement suivant : `~/DO280/labs/network-review/php-http.yaml`.  
Apportez les modifications nécessaires au fichier, utilisez l'image de conteneur `quay.io/redhattraining/php-ssl:v1.0` et définissez le port sur 8080.  
Après avoir créé l'application, attendez quelques instants pour vous assurer qu'un pod est en cours d'exécution.
3. Créez une route nommée `php-http`, avec le nom d'hôte `php-http.apps.ocp4.example.com`, pour accéder à l'application.

Sur la machine **workstation**, utilisez Firefox pour accéder à l'URL d'application :

- <http://php-http.apps.ocp4.example.com/>.

Confirmez la disponibilité de l'application avant de passer à l'étape suivante.

4. Créez une politique réseau dans l'espace de noms **network-review** pour refuser tout le trafic entrant par défaut. Lorsqu'elle est correctement configurée, la politique réseau empêche également les pods de communiquer entre eux au sein de l'espace de noms **network-review**. Utilisez le fichier de ressources disponible à l'emplacement suivant : `~/D0280/labs/network-review/deny-all.yaml`. Effectuez les modifications nécessaires pour cibler tous les pods de l'espace de noms.
5. Créez une politique réseau pour autoriser le trafic entrant vers les routes de l'espace de noms **network-review**. Utilisez le fichier de ressources disponible à l'emplacement suivant : `~/D0280/labs/network-review/allow-from-openshift-ingress.yaml`. Effectuez les modifications nécessaires pour cibler tous les pods de l'espace de noms et autoriser le trafic en provenance du contrôleur d'entrée par défaut. Étant donné que l'environnement de formation utilise la stratégie de point d'accès **HostNetwork**, attribuez le libellé **default** à l'espace de noms `network.openshift.io/policy-group=ingress`. Cette action doit être exécutée en tant qu'utilisateur **admin**.
6. En tant qu'utilisateur **developer**, générez et signez un certificat TLS pour la version chiffrée de l'application. Générez une demande de signature de certificat (CSR) pour le nom d'hôte `php-https.apps.ocp4.example.com`. Enregistrez le CSR dans `~/D0280/labs/network-review/certs/training.csr`. Utilisez le CSR pour générer un certificat et enregistrez-le dans `~/D0280/labs/network-review/certs/training.crt`. Pour générer le certificat, transmettez comme arguments le certificat d'autorité de certification accessible à l'emplacement `~/D0280/labs/network-review/certs/training-CA.pem` et le CSR. Vous pouvez utiliser le fichier texte `~/D0280/labs/network-review/certs/openssl-commands.txt` pour obtenir de l'aide. Ce fichier contient les commandes permettant de générer la demande de signature de certificat et le certificat. Veillez à remplacer les valeurs du fichier avant de copier et d'exécuter les commandes OpenSSL.
7. Créez un secret TLS OpenShift nommé **php-certs** dans le projet **network-review**. Utilisez le fichier `~/D0280/labs/network-review/certs/training.crt` pour le certificat et le fichier `~/D0280/labs/network-review/certs/training.key` pour la clé.
8. Utilisez le fichier de ressources, disponible à l'emplacement `~/D0280/labs/network-review/php-https.yaml`, pour déployer la version sécurisée de l'application PHP. Déployez l'application dans le projet **network-review**. Avant de déployer l'application, apportez les modifications nécessaires au fichier de ressources, à savoir :
  - l'emplacement du conteneur ;
  - le port sur lequel l'application écoute ;
  - le nom du secret à monter en tant que volume.

- 9.** Créez une route directe sécurisée nommée `php-https`, avec le nom d'hôte `php-https.apps.ocp4.example.com`, pour accéder à la version sécurisée de l'application.

Sur la machine `workstation`, utilisez Firefox pour accéder à l'URL d'application :

- `https://php-https.apps.ocp4.example.com/`.

Acceptez le certificat signé et confirmez la disponibilité de l'application.

- 10. Étape facultative** : sur la machine `workstation`, utilisez la commande `curl` pour accéder à la version HTTPS de l'application.

Transmettez le certificat d'autorité de certification à la commande `curl` pour valider la connexion sécurisée.

- 11.** Revenez au répertoire personnel.

```
[student@workstation network-review]$ cd
```

## Évaluation

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab network-review grade
```

## Fin

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour effectuer cet exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab network-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Configuration de la mise en réseau OpenShift pour les applications

Dans cet atelier, vous allez configurer une route directe TLS pour votre application.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Déployer une application et configurer une route non sécurisée.
- Limiter le trafic vers les applications.
- Générer un certificat TLS pour une application.
- Configurer une route directe pour une application avec un certificat TLS.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée l'autorité de certification autosignée qui sera utilisée dans cet atelier.

```
[student@workstation ~]$ lab network-review start
```

## Instructions

Dans cette révision, vous déployez une application PHP qui imprime des informations sur le système. L'application est disponible selon deux configurations différentes : la première fonctionne avec un réseau non chiffré qui écoute sur le port 8080 et la deuxième utilise un certificat TLS pour chiffrer le trafic réseau, qui écoute sur le port 8443.

L'image de conteneur pour cette révision est accessible à l'adresse `quay.io/redhattraining/php-ss1`. Elle comporte deux balises : `v1.0` pour la version non sécurisée de l'application et `v1.1` pour la version sécurisée.

1. En tant qu'utilisateur `developer` d'Openshift, créez le projet `network-review`.

1.1. Connectez-vous au cluster en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

1.2. Créez le projet `network-review`.

```
[student@workstation ~]$ oc new-project network-review
Now using project "network-review" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

2. En tant qu'utilisateur **developer**, déployez la version non sécurisée de l'application PHP dans le projet **network-review**. Utilisez le fichier de ressources disponible à l'emplacement suivant : `~/D0280/labs/network-review/php-http.yaml`.

Apportez les modifications nécessaires au fichier, utilisez l'image de conteneur `quay.io/redhattraining/php-ssl:v1.0` et définissez le port sur 8080.

Après avoir créé l'application, attendez quelques instants pour vous assurer qu'un pod est en cours d'exécution.

- 2.1. Accédez au répertoire `~/D0280/labs/network-review`.

```
[student@workstation ~]$ cd ~/D0280/labs/network-review
```

- 2.2. Utilisez un éditeur de texte pour mettre à jour le fichier `php-http.yaml` comme suit :

- Définissez l'image de sorte que l'image de conteneur accessible à l'adresse `quay.io/redhattraining/php-ssl:v1.0` soit utilisée.
- Identifiez l'entrée `containerPort` et définissez la valeur sur `8080`, qui correspond au point d'accès **non sécurisé**.

```
...output omitted...
spec:
...output omitted...
template:
...output omitted...
spec:
  containers:
...output omitted...
    image: 'quay.io/redhattraining/php-ssl:v1.0'
    name: php-http
    ports:
      - containerPort: 8080
        name: php-http
...output omitted...
```

Après avoir effectué vos modifications, enregistrez le fichier et fermez-le.

- 2.3. Utilisez la commande `oc create` pour déployer l'application. Cela permet de créer un déploiement et un service.

```
[student@workstation network-review]$ oc create -f php-http.yaml
deployment.apps/php-http created
service/php-http created
```

- 2.4. Attendez quelques instants, puis exécutez la commande `oc get pods` pour vous assurer qu'un pod est en cours d'exécution.

```
[student@workstation network-review]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
php-http-6cb58c847b-7qsbd  1/1     Running   0          8m11s
```

3. Créez une route nommée `php-http`, avec le nom d'hôte `php-http.apps.ocp4.example.com`, pour accéder à l'application.

Sur la machine `workstation`, utilisez Firefox pour accéder à l'URL d'application :

- `http://php-http.apps.ocp4.example.com/`.

Confirmez la disponibilité de l'application avant de passer à l'étape suivante.

- 3.1. Exécutez la commande `oc expose` pour créer une route permettant d'accéder à l'application. Attribuez à la route le nom d'hôte `php-http.apps.ocp4.example.com`.

```
[student@workstation network-review]$ oc expose svc php-http \
>   --hostname php-http.apps.ocp4.example.com
route.route.openshift.io/php-http exposed
```

- 3.2. Récupérez le nom de la route et copiez-le dans le presse-papiers.

```
[student@workstation network-review]$ oc get routes
NAME      HOST/PORT          PATH  SERVICES  PORT  ...
php-http  php-http.apps.ocp4.example.com  php-http  8080  ...
```

- 3.3. Sur la machine `workstation`, ouvrez Firefox et accédez à l'URL de l'application :

- `http://php-http.apps.ocp4.example.com/`.

Vérifiez que vous pouvez voir l'application.

## About this application

### **⚠ The application is currently served over HTTP**

- **Current system load:** 2.5
- **Number of connections:** 1
- **Memory usage:** 8 Mb

4. Créez une politique réseau dans l'espace de noms `network-review` pour refuser tout le trafic entrant par défaut. Lorsqu'elle est correctement configurée, la politique réseau empêche également les pods de communiquer entre eux au sein de l'espace de noms `network-review`.

Utilisez le fichier de ressources disponible à l'emplacement suivant : `~/D0280/labs/network-review/deny-all.yaml`. Effectuez les modifications nécessaires pour cibler tous les pods de l'espace de noms.

- 4.1. Utilisez un éditeur de texte pour mettre à jour le fichier `deny-all.yaml` avec un sélecteur de pod vide afin de cibler tous les pods de l'espace de noms.

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: deny-all
spec:
  podSelector: {}
```

- 4.2. Créez la politique réseau à l'aide de la commande `oc create`.

```
[student@workstation network-review]$ oc create -f deny-all.yaml
networkpolicy.networking.k8s.io/deny-all created
```

- 4.3. Utilisez la commande `curl` pour vérifier qu'il n'y a aucun accès au pod `php-http` à partir de la route. Attendez quelques secondes, puis appuyez sur `Ctrl+C` pour quitter la commande `curl`.

```
[student@workstation network-review]$ curl http://php-http.apps.ocp4.example.com
^C
```



### Important

Si le pod `php-http` est arrivé sur le même nœud qu'un pod `router-default`, la commande `curl` fonctionne lorsque le trafic transite par ce pod de routeur. C'est uniquement le cas avec les clusters à trois noeuds. Dans un cluster OpenShift traditionnel, dans lequel les noeuds d'infrastructure ou de plan de contrôle sont séparés des noeuds de calcul, la politique réseau s'applique à tous les pods de routeur du cluster.

Si la commande `curl` réussit, exécutez-la à nouveau pour confirmer que la politique réseau fonctionne comme prévu. Cette seconde tentative doit transiter par l'autre pod de routeur dans le cluster.

5. Créez une politique réseau pour autoriser le trafic entrant vers les routes de l'espace de noms `network-review`.

Utilisez le fichier de ressources disponible à l'emplacement suivant : `~/DO280/labs/network-review/allow-from-openshift-ingress.yaml`. Effectuez les modifications nécessaires pour cibler tous les pods de l'espace de noms et autoriser le trafic en provenance du contrôleur d'entrée par défaut.

Étant donné que l'environnement de formation utilise la stratégie de point d'accès `HostNetwork`, attribuez le libellé `default` à l'espace de noms `network.openshift.io/policy-group=ingress`. Cette action doit être exécutée en tant qu'utilisateur `admin`.

- 5.1. Utilisez un éditeur de texte pour mettre à jour le fichier `allow-from-openshift-ingress.yaml` avec un sélecteur de pod vide afin de cibler tous les pods de l'espace de noms. Incluez un sélecteur d'espace de noms pour établir une correspondance avec le libellé `network.openshift.io/policy-group=ingress`.

```
...output omitted...
spec:
  podSelector: {}
  ingress:
    - from:
      - namespaceSelector:
          matchLabels:
            network.openshift.io/policy-group: ingress
```

- 5.2. Créez la politique réseau à l'aide de la commande `oc create`.

```
[student@workstation network-review]$ oc create -f \
>   allow-from-openshift-ingress.yaml
networkpolicy.networking.k8s.io/allow-from-openshift-ingress created
```

- 5.3. Connectez-vous en tant qu'utilisateur `admin`.

```
[student@workstation network-review]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 5.4. Appliquez le libellé `network.openshift.io/policy-group=ingress` à l'espace de noms `default`.

```
[student@workstation network-policy]$ oc label namespace default \
>   network.openshift.io/policy-group=ingress
namespace/default labeled
```

- 5.5. Utilisez la commande `curl` pour vérifier qu'il y a un accès au pod `php-http` à partir de la route. Étant donné qu'un cluster à trois nœuds est exécuté dans la salle de classe, exécutez la commande `curl` à plusieurs reprises pour valider l'accès via tous les pods de routeur.

```
[student@workstation network-review]$ for X in {1..4}
>   do
>     curl -s http://php-http.apps.ocp4.example.com | grep "PHP"
>   done
<title>PHP Application</title>
<title>PHP Application</title>
<title>PHP Application</title>
<title>PHP Application</title>
```

6. En tant qu'utilisateur `developer`, générez et signez un certificat TLS pour la version chiffrée de l'application.

Générez une demande de signature de certificat (CSR) pour le nom d'hôte `php-https.apps.ocp4.example.com`. Enregistrez le CSR dans `~/D0280/labs/network-review/certs/training.csr`.

Utilisez le CSR pour générer un certificat et enregistrez-le dans `~/D0280/labs/network-review/certs/training.crt`. Pour générer le certificat, transmettez comme arguments

**chapitre 12 |** Configuration de la mise en réseau OpenShift pour les applications

le certificat d'autorité de certification accessible à l'emplacement ~/D0280/labs/network-review/certs/training-CA.pem et le CSR.

Vous pouvez utiliser le fichier texte ~/D0280/labs/network-review/certs/openssl-commands.txt pour obtenir de l'aide. Ce fichier contient les commandes permettant de générer la demande de signature de certificat et le certificat. Veillez à remplacer les valeurs du fichier avant de copier et d'exécuter les commandes OpenSSL.

- 6.1. Connectez-vous en tant qu'utilisateur developer pour effectuer le reste de cet atelier.

```
[student@workstation network-review]$ oc login -u developer -p developer  
Login successful.
```

*...output omitted...*

- 6.2. Accédez au répertoire ~/D0280/labs/network-review/certs.

```
[student@workstation network-review]$ cd certs
```

- 6.3. Générez la demande de signature de certificat (CSR) pour php-<https://apps.ocp4.example.com>. Veillez à saisir l'objet de la demande sur une seule ligne. Sinon, supprimez l'option -subj et son contenu. Cette commande vous invite à saisir des valeurs ; veillez à indiquer un nom courant (CN) php-<https://apps.ocp4.example.com>.

**Note**

Assurez-vous qu'il n'y a pas d'espace après la barre oblique de fin de l'organisation (Red Hat) et le nom commun (CN).

```
[student@workstation certs]$ openssl req -new -key training.key \  
> -subj "/C=US/ST=North Carolina/L=Raleigh/O=Red Hat/\\  
> CN=php-https://apps.ocp4.example.com" \  
> -out training.csr
```

Sinon, ouvrez le fichier texte openssl-commands.txt. Copiez et collez la première commande openssl sur votre terminal. Remplacez le domaine générique par [apps.ocp4.example.com](https://apps.ocp4.example.com) et le fichier de sortie par training.csr.

**Note**

Cette commande ne génère aucune sortie.

- 6.4. Générez le certificat signé. Notez l'utilisation des options -CA et -CAkey pour signer le certificat avec l'autorité de certification.

```
[student@workstation certs]$ openssl x509 -req -in training.csr \
> -CA training-CA.pem -CAkey training-CA.key -CAcreateserial \
> -passin file:passphrase.txt \
> -out training.crt -days 3650 -sha256 -extfile training.ext
Signature ok
subject=C = US, ST = North Carolina, L = Raleigh, O = Red Hat, CN = php-
https.apps.ocp4.example.com
Getting CA Private Key
```

Vous pouvez également copier et coller la deuxième commande `openssl` dans le fichier `openssl-commands.txt` sur votre terminal. Remplacez le fichier CSR par `training.csr`, l'autorité de certification par `training-CA.pem` et le certificat de sortie par `training.crt`.

- 6.5. Assurez-vous que le certificat et la clé que vous venez de créer se trouvent dans le répertoire actif.

```
[student@workstation certs]$ ls -l
total 36
-rw-rw-r-- 1 student student 566 abr 26 07:43 openssl-commands.txt
-rw-rw-r-- 1 student student 33 jun 15 22:20 passphrase.txt
-rw----- 1 student student 1743 jun 15 22:20 training-CA.key
-rw-r--r-- 1 student student 1334 jun 15 22:20 training-CA.pem
-rw-rw-r-- 1 student student 41 jun 15 22:33 training-CA.srl
-rw-rw-r-- 1 student student 1395 jun 15 22:33 training.crt
-rw-rw-r-- 1 student student 1021 jun 15 22:33 training.csr
-rw-r--r-- 1 student student 352 jun 15 22:20 training.ext
-rw----- 1 student student 1679 jun 15 22:20 training.key
```

- 6.6. Revenez au répertoire `network-review`. Cette étape est importante, car la suivante implique la création d'un itinéraire à l'aide du certificat signé.

```
[student@workstation certs]$ cd ~/D0280/labs/network-review
```

7. Créez un secret TLS OpenShift nommé `php-certs` dans le projet `network-review`. Utilisez le fichier `~/D0280/labs/network-review/certs/training.crt` pour le certificat et le fichier `~/D0280/labs/network-review/certs/training.key` pour la clé.
  - 7.1. Utilisez la commande `oc create secret` pour créer le secret TLS `php-certs`. Transmettez le fichier `training.crt` comme certificat et `training.key` comme clé.

```
[student@workstation network-review]$ oc create secret tls php-certs \
> --cert certs/training.crt --key certs/training.key
secret/php-certs created
```

- 7.2. Récupérez la liste des secrets pour vous assurer qu'elle est bien présente.

```
[student@workstation network-review]$ oc get secrets
NAME          TYPE           DATA   AGE
...output omitted...
php-certs    kubernetes.io/tls      2      93s
```

8. Utilisez le fichier de ressources, disponible à l'emplacement ~/D0280/Labs/network-review/php-https.yaml, pour déployer la version sécurisée de l'application PHP. Déployez l'application dans le projet **network-review**.

Avant de déployer l'application, apportez les modifications nécessaires au fichier de ressources, à savoir :

- l'emplacement du conteneur ;
- le port sur lequel l'application écoute ;
- le nom du secret à monter en tant que volume.

- 8.1. Utilisez un éditeur de texte pour mettre à jour le fichier **php-https.yaml** comme suit :

- Définissez l'image de sorte que l'image de conteneur accessible à l'adresse **quay.io/redhattraining/php-ssl:v1.1** soit utilisée.
- Identifiez l'entrée **containerPort** et définissez la valeur sur **8443**, qui correspond au point d'accès **sécurisé**.
- Identifiez l'entrée **secretName** et définissez la valeur sur **php-certs**, qui correspond au nom du secret que vous avez créé lors d'une étape précédente.

```
...output omitted...
spec:
...output omitted...
  template:
...output omitted...
    spec:
      containers:
...output omitted...
      image: 'quay.io/redhattraining/php-ssl:v1.1'
      name: php-https
      ports:
        - containerPort: 8443
          name: php-https
...output omitted...
      volumes:
        - name: tls-certs
          secret:
            secretName: php-certs
```

Après avoir effectué vos modifications, enregistrez le fichier et fermez-le.

- 8.2. Utilisez la commande **oc create** pour déployer l'application sécurisée. Cela permet de créer un déploiement et un service.

```
[student@workstation network-review]$ oc create -f php-https.yaml
deployment.apps/php-https created
service/php-https created
```

- 8.3. Attendez quelques instants, puis exécutez la commande `oc get pods` pour vous assurer que le pod `php-https` est en cours d'exécution.

```
[student@workstation network-review]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
php-http-6cb58c847b-7qsbd  1/1     Running   0          8m11s
php-https-84498cd794-hvf7w 1/1     Running   0          26s
```

9. Créez une route directe sécurisée nommée `php-https`, avec le nom d'hôte `php-https.apps.ocp4.example.com`, pour accéder à la version sécurisée de l'application. Sur la machine `workstation`, utilisez Firefox pour accéder à l'URL d'application :

- `https://php-https.apps.ocp4.example.com/`.

Acceptez le certificat signé et confirmez la disponibilité de l'application.

- 9.1. Exécutez la commande `oc create route` pour créer une route directe permettant d'accéder à l'application. Attribuez à la route le nom d'hôte `php-https.apps.ocp4.example.com`. Utilisez l'option `port` pour indiquer le port sécurisé 8443.

```
[student@workstation network-review]$ oc create route passthrough php-https \
>   --service php-https --port 8443 --hostname php-https.apps.ocp4.example.com
route.route.openshift.io/php-https created
```

- 9.2. Récupérez le nom de la route et copiez-le dans le presse-papiers.

```
[student@workstation network-review]$ oc get routes
NAME      HOST/PORT            ... SERVICES   PORT  TERMINATION
php-http  php-http.apps.ocp4.example.com  ...  php-http  8080
php-https  php-https.apps.ocp4.example.com  ...  php-https  8443  passthrough
```

- 9.3. Sur la machine `workstation`, ouvrez Firefox et accédez à l'URL de l'application :

- `https://php-https.apps.ocp4.example.com/`.

Acceptez le certificat signé et confirmez que vous pouvez voir la version sécurisée de l'application.

## About this application

### 🔒 The application is currently served over TLS

- **Current system load:** 1
- **Number of connections:** 0
- **Memory usage:** 8 Mb

10. **Étape facultative** : sur la machine `workstation`, utilisez la commande `curl` pour accéder à la version HTTPS de l'application.

Transmettez le certificat d'autorité de certification à la commande `curl` pour valider la connexion sécurisée.

Utilisez l'option `--cacert` pour transmettre le certificat d'autorité de certification à la commande `curl`.

```
[student@workstation network-review]$ curl -v --cacert certs/training-CA.pem \
>   https://php-https.apps.ocp4.example.com
...output omitted...
* Server certificate:
*   subject: C=US; ST=North Carolina; L=Raleigh; O=Red Hat; \
CN=php-https.apps.ocp4.example.com
...output omitted...
*   SSL certificate verify ok.
...output omitted...
```

11. Revenez au répertoire personnel.

```
[student@workstation network-review]$ cd
```

## Évaluation

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab network-review grade
```

## Fin

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour effectuer cet exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab network-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- OpenShift met en œuvre un SDN (réseau défini par logiciel) pour gérer l'infrastructure réseau du cluster. Le SDN dissocie le logiciel qui gère le trafic des mécanismes sous-jacents qui acheminent le trafic.
- Kubernetes fournit des services qui permettent de regrouper logiquement les pods sous un itinéraire d'accès commun. Les services agissent comme des modules d'équilibrage de charge devant un ou plusieurs pods.
- Les services utilisent des sélecteurs (libellés) qui indiquent les pods disponibles pour le service.
- Il existe deux types de routes : sécurisées et non sécurisées. Les routes sécurisées chiffrent le trafic à l'aide de certificats TLS, tandis que les routes non sécurisées le transfèrent via une connexion non chiffrée.

Les routes sécurisées prennent en charge trois modes : périphérie, direct et rechiffrement.

- Les politiques réseau contrôlent le trafic réseau vers les pods. Des zones logiques peuvent être créées dans le SDN pour séparer le trafic entre les pods de n'importe quel espace de noms.



## chapitre 13

# Contrôle de la planification des pods

### Objectif

Contrôler les nœuds sur lesquels s'exécute un pod.

### Résultats

- Décrire les algorithmes de planification des pods, les méthodes utilisées pour influencer la planification et appliquer ces méthodes.
- Limiter les ressources utilisées par les conteneurs, les pods et les projets.
- Contrôler le nombre de réplicas d'un pod, spécifier le nombre de réplicas dans un déploiement, mettre à l'échelle manuellement le nombre de réplicas et créer une ressource de mise à l'échelle automatique de pod horizontale (HPA).

### Sections

- Contrôle du comportement de planification des pods (et exercice guidé)
- Limitation de l'utilisation des ressources par une application (et exercice guidé)
- Évolutivité d'une application (et exercice guidé)

### Atelier

Contrôle de la planification des pods

# Contrôle du comportement de planification des pods

---

## Résultats

Après avoir terminé cette section, vous devez pouvoir décrire les algorithmes de planification des pods et les méthodes utilisées pour influencer la planification, et appliquer ces méthodes.

## Présentation de l'algorithme du planificateur OpenShift

Le planificateur de pods détermine le placement des nouveaux pods sur les nœuds dans le cluster OpenShift. Il est également conçu pour être hautement configurable et adaptable à différents clusters. La configuration par défaut à la livraison de Red Hat OpenShift Container Platform prend en charge les concepts de datacenters communs de zones et de régions en utilisant des libellés de nœud, des règles d'affinité et des règles d'anti-affinité.

L'algorithme du planificateur de pods OpenShift suit un processus en trois étapes :

1. Filtrage des nœuds.

Le planificateur filtre la liste des nœuds en cours d'exécution en évaluant chaque nœud par rapport à un ensemble de prédictats, tels que la disponibilité des ports hôte, ou si un pod peut être planifié sur un nœud dont le disque ou la mémoire est sollicité.

Un pod peut également définir un sélecteur de nœud qui correspond aux libellés dans les nœuds du cluster. Les nœuds dont les libellés ne correspondent pas ne sont pas éligibles.

Un pod peut également définir les demandes de ressources pour les ressources informatiques telles que le processeur, la mémoire et le stockage. Les nœuds disposant de ressources informatiques insuffisantes ne sont pas éligibles.

Un autre contrôle de filtrage permet d'évaluer si la liste des nœuds présente des rejets et, si tel est le cas, si le pod a une tolérance associée qui peut accepter le rejet. Si un pod ne peut pas accepter le rejet d'un nœud, le nœud n'est pas éligible. Par défaut, les nœuds de plan de contrôle incluent le rejet `node-role.kubernetes.io/master:NoSchedule`. Un pod qui n'a aucune tolérance correspondante pour ce rejet ne sera pas planifié sur un nœud de plan de contrôle.



### Note

L'environnement de formation utilise un cluster à trois nœuds qui n'inclut pas de nœuds de calcul supplémentaires. Le cluster à trois nœuds est disponible pour une installation de système nu dans OpenShift Container Platform 4.10. Ce type de cluster s'applique aux environnements soumis à des contraintes de ressources, tels que les déploiements périphériques éloignés.

Les nœuds de plan de contrôle d'un cluster à trois nœuds n'incluent pas le rejet `node-role.kubernetes.io/master:NoSchedule`. Les pods d'application ordinaires peuvent être planifiés sur les nœuds de plan de contrôle.

L'étape de filtrage aboutit généralement à une liste raccourcie de candidats de nœuds qui sont habilités à exécuter le pod. Dans certains cas, aucun des nœuds n'est filtré, ce qui signifie que le pod peut s'exécuter sur n'importe quel nœud. Dans d'autres cas, tous les nœuds sont filtrés, ce qui signifie que le pod ne peut pas être planifié tant qu'un nœud disposant des conditions préalables requises n'est pas disponible.

Vous trouverez une liste complète des prédictats et leurs descriptions dans la section Références.

## 2. Définition de priorités dans la liste de nœuds filtrée.

La liste de nœuds candidats est évaluée en fonction de plusieurs critères de priorité, qui s'ajoutent à une note pondérée. Les nœuds ayant les valeurs les plus élevées sont les meilleurs candidats pour exécuter le pod.

Parmi les critères se trouvent les règles **affinity** et **anti-affinity**. Les nœuds ayant une affinité plus élevée avec le pod ont une note supérieure, et les nœuds ayant une anti-affinité plus élevée ont une note inférieure.

Les règles **affinity** sont couramment utilisées pour planifier les pods associés de sorte qu'ils soient proches les uns des autres pour des raisons de performances. Cela implique, par exemple, d'utiliser le même système réseau dorsal pour les pods qui doivent être synchronisés entre eux.

Les règles **anti-affinity** sont couramment utilisées pour planifier les pods associés de sorte qu'ils ne soient pas trop proches les uns des autres dans un souci de haute disponibilité. Cela implique, par exemple, d'éviter de planifier tous les pods d'une même application sur le même nœud.

## 3. Sélection du nœud le plus adapté.

La liste de candidats est triée en fonction de ces notes ; le nœud ayant la note la plus élevée est sélectionné pour héberger le pod. Si plusieurs nœuds ont la même note élevée, l'un d'eux est sélectionné à tour de rôle.

Le planificateur est flexible et peut être personnalisé afin de répondre à des situations de planification avancées. De plus, bien que ce cours se concentre sur le placement des pods à l'aide de libellés de nœud et de sélecteurs de nœud, les pods peuvent également être placés à l'aide des règles d'affinité et d'anti-affinité des pods, ainsi que des règles d'affinité et d'anti-affinité des nœuds. La personnalisation du planificateur et la présentation de ces autres scénarios de placement des pods ne sont pas abordées dans ce cours.

# Planification et topologie

Une topologie courante pour les grands data centers, comme les fournisseurs de cloud, consiste à organiser les hôtes en **regions** et en **zones** :

- Une **region** est un ensemble d'hôtes dans une zone géographique, ce qui garantit une connectivité haut débit entre eux.
- Une **zone**, également appelée **availability zone**, est un ensemble d'hôtes qui risquent d'échouer ensemble, car ils partagent des composants d'infrastructure critiques communs, comme un commutateur réseau, une baie de stockage ou un onduleur (UPS).

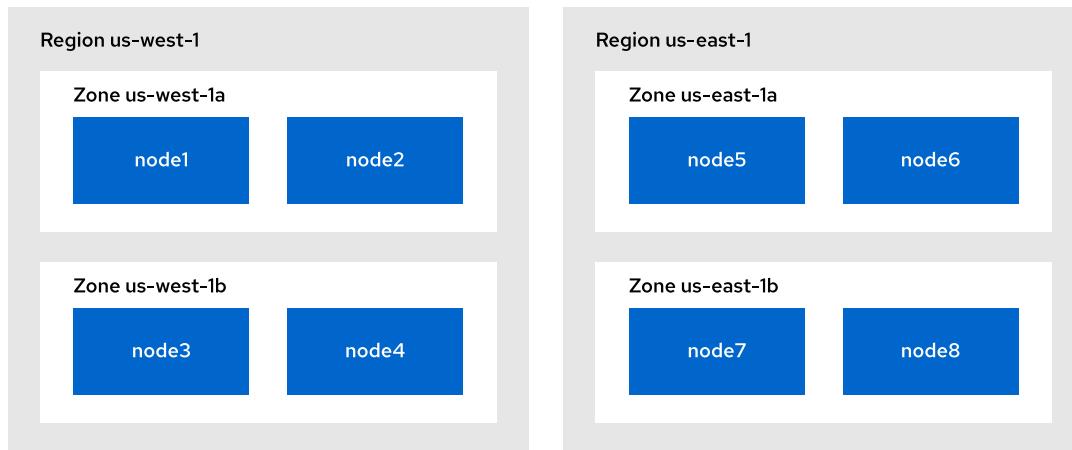
À titre d'exemple de régions et de zones, Amazon Web Services (AWS) dispose d'une région en Virginie du Nord (`us-east-1`) avec 6 zones de disponibilité et une autre région dans l'Ohio (`us-`

east -2) avec 3 zones de disponibilité. Chacune des zones de disponibilité AWS peut contenir plusieurs datacenters pouvant être constitués de centaines de milliers de serveurs.

La configuration standard du planificateur de pods OpenShift prend en charge ce type de topologie de cluster en définissant des prédictats dans les libellés **region** et **zone**. Les prédictats sont définis de sorte que :

- Les répliques de pods, créés à partir du même déploiement sont planifiés pour être exécutés sur des nœuds ayant la même valeur que le libellé **region**.
- Les répliques de pods sont planifiées pour s'exécuter sur des nœuds ayant des valeurs différentes pour le libellé **zone**.

La figure suivante illustre un exemple de topologie composée de plusieurs régions comptant chacune différentes zones composées à leur tour de plusieurs nœuds.



**Figure 13.1: Exemple de topologie de cluster utilisant des régions et des zones**

## Étiquetage des nœuds

En tant qu'administrateur de cluster OpenShift, vous pouvez ajouter des libellés supplémentaires à vos nœuds. Par exemple, vous pouvez étiqueter les nœuds avec le libellé **env** en utilisant les valeurs de **dev**, **qa** ou **prod** dans le but de déployer les charges de travail de développement, d'assurance-qualité et de production sur un sous-ensemble de nœuds spécifique. Les libellés que vous choisissez sont arbitraires, mais vous devez communiquer les libellés et les valeurs qui y sont associées à vos développeurs pour qu'ils puissent configurer leurs applications de manière appropriée.

Utilisez la commande `oc label` en tant qu'administrateur de cluster pour ajouter, mettre à jour ou supprimer immédiatement un libellé de nœud. Par exemple, utilisez la commande suivante pour étiqueter un nœud avec `env=dev` :

```
[user@host ~]$ oc label node master01 env=dev
```

Utilisez l'option `--overwrite` pour modifier un libellé existant :

```
[user@host ~]$ oc label node master01 env=prod --overwrite
```

Supprimez un libellé en spécifiant le nom du libellé suivi d'un tiret, comme `env-` :

```
[user@host ~]$ oc label node master01 env-
```

**Important**

Les libellés et leurs valeurs sont sensibles à la casse. Un sélecteur de nœud d'application doit correspondre à la casse du libellé réel et à la valeur appliquée au nœud.

Utilisez l'option `--show-labels` avec la commande `oc get nodes` pour afficher les libellés sensibles à la casse qui sont affectés à un nœud :

```
[user@host ~]$ oc get node master02 --show-labels

NAME      ... ROLES      ... LABELS
master02 ... master,worker ... beta.kubernetes.io/arch=amd64,beta.kubernetes.io/
os=linux,kubernetes.io/arch=amd64,kubernetes.io/hostname=master02,kubernetes.io/
os=linux,node-role.kubernetes.io/master=,node-role.kubernetes.io/
worker=,node.openshift.io/os_id=rhcos
```

Notez qu'un nœud peut se voir affecter plusieurs libellés par défaut par OpenShift. Les libellés dont les clés incluent le suffixe `kubernetes.io` ne doivent pas être modifiés par un administrateur de cluster, car ils sont utilisés en interne par le planificateur. Les nœuds affichés dans ces exemples de commandes utilisent la configuration d'automatisation de pile complète AWS.

Les administrateurs de cluster peuvent également utiliser l'option `-L` pour déterminer la valeur d'un libellé unique. Par exemple :

```
[user@host ~]$ oc get node -L failure-domain.beta.kubernetes.io/region
NAME      STATUS    ROLES      AGE      VERSION      REGION
master01  Ready     master,worker  42d     v1.23.3+e419edf
master02  Ready     master,worker  42d     v1.23.3+e419edf
master03  Ready     master,worker  42d     v1.23.3+e419edf
```

Plusieurs options `-L` sont prises en charge dans la même commande `oc get`. Par exemple :

```
[user@host ~]$ oc get node -L failure-domain.beta.kubernetes.io/region \
>   -L failure-domain.beta.kubernetes.io/zone -L env
NAME      STATUS    ROLES      AGE      VERSION      ... ENV
master01  Ready     master,worker  42d     v1.23.3+e419edf
master02  Ready     master,worker  42d     v1.23.3+e419edf
master03  Ready     master,worker  42d     v1.23.3+e419edf
```

## Étiquetage des ensembles de machines

Bien que les libellés de nœud soient persistants, si votre cluster OpenShift contient des ensembles de machines, vous devez également ajouter des libellés à la configuration de l'ensemble de machines. Cela permet de s'assurer que les nouvelles machines (et les nœuds générés à partir de celles-ci) contiendront également les libellés souhaités. Les ensembles de machines se trouvent dans les clusters qui utilisent l'automatisation de pile complète et dans certains clusters qui

**chapitre 13 |** Contrôle de la planification des pods

utilisent une infrastructure préexistante permettant l'intégration du fournisseur de cloud. Les clusters de systèmes nus n'utilisent pas d'ensembles de machines.

Vous pouvez identifier la relation entre les machines et les nœuds en listant les machines dans l'espace de noms `openshift-machine-api` et en incluant l'option `-o wide`:

```
[user@host ~]$ oc get machines -n openshift-machine-api -o wide
NAME          ... NODE
...output omitted...
ocp-qz7hf-worker-us-west-1b-rvx6w ... ip-10-0-139-250.us-west-1.compute.internal
ocp-qz7hf-worker-us-west-1c-v4n4n ... ip-10-0-154-226.us-west-1.compute.internal
```

Les machines utilisées pour les nœuds `worker` doivent provenir d'un ensemble de machines. Le nom d'une machine contient le nom de l'ensemble de machines duquel elle a été générée. Utilisez la commande suivante pour lister les ensembles de machines :

```
[user@host ~]$ oc get machineset -n openshift-machine-api
NAME        DESIRED  CURRENT  READY  AVAILABLE ...
ocp-qz7hf-worker-us-west-1b  1        1        1        1        ...
ocp-qz7hf-worker-us-west-1c  1        1        1        1        ...
```

Modifiez un ensemble de machines afin que les nouvelles machines qui y sont générées disposent du libellé ou des libellés souhaités. Le fait de modifier un ensemble de machines n'entraînera pas de modifications sur les machines ou les nœuds existants. Utilisez la commande suivante pour modifier un ensemble de machines :

```
[user@host ~]$ oc edit machineset ocp-qz7hf-worker-us-west-1b \
>   -n openshift-machine-api
```

Les lignes mises en surbrillance ci-dessous indiquent où ajouter un libellé dans un ensemble de machines :

```
...output omitted...
spec:
  metadata:
    creationTimestamp: null
  labels:
    env: dev
  providerSpec:
...output omitted...
```

## Contrôle du placement des pods

De nombreux pods liés à l'infrastructure d'un cluster OpenShift sont configurés pour s'exécuter sur des nœuds de plan de contrôle. Il s'agit, par exemple, des pods pour l'opérateur DNS, l'opérateur OAuth et le serveur d'API OpenShift. Dans certains cas, il faut utiliser pour cela le sélecteur de nœud `node-role.kubernetes.io/master: ''` dans la configuration d'un ensemble de démons ou d'un ensemble de répliques.

De même, il se peut que certaines applications utilisateur doivent s'exécuter sur un ensemble de nœuds spécifique. Par exemple, certains nœuds offrent l'accélération matérielle pour certains types de charges de travail, ou l'administrateur de cluster ne veut pas mélanger applications de

production et applications de développement. Utilisez les libellés de nœuds et les sélecteurs de nœuds pour mettre en œuvre ce genre de scénarios.

Un sélecteur de nœud relève de la définition d'un pod individuel. Définissez un sélecteur de nœud dans une ressource de déploiement, de sorte que tout nouveau pod généré à partir de cette ressource dispose du sélecteur de nœud souhaité. Si votre ressource de déploiement est sous contrôle de version, modifiez le fichier de ressources et appliquez les modifications à l'aide de la commande `oc apply -f`.

Il est également possible d'ajouter ou de modifier un sélecteur de nœud à l'aide de la commande `oc edit` ou de la commande `oc patch`. Par exemple, pour configurer le déploiement `myapp` de sorte que ses pods s'exécutent uniquement sur des nœuds ayant le libellé `env=qa`, utilisez la commande `oc edit`:

```
[user@host ~]$ oc edit deployment/myapp

...
spec:
  ...
  template:
    metadata:
      annotations:
        openshift.io/generated-by: OpenShiftNewApp
      creationTimestamp: null
      labels:
        deployment: myapp
    spec:
      nodeSelector:
        env: dev
      containers:
        - image: quay.io/redhattraining/scaling:v1.0
...

```

La commande `oc patch` suivante effectue la même chose :

```
[user@host ~]$ oc patch deployment/myapp --patch \
> '{"spec": {"template": {"spec": {"nodeSelector": {"env": "dev"}}}}}'
```

Que vous utilisez la commande `oc edit` ou la commande `oc patch`, la modification déclenche un nouveau déploiement et les nouveaux pods sont planifiés en fonction du sélecteur de nœud.

## Configuration d'un sélecteur de nœud pour un projet

Si l'administrateur de cluster ne veut pas que les développeurs contrôlent le sélecteur de nœud pour leurs pods, un sélecteur de nœud par défaut doit être configuré dans la ressource du projet. Un administrateur de cluster peut soit définir un sélecteur de nœud lors de la création d'un projet, soit ajouter ou mettre à jour un sélecteur de nœud après la création d'un projet. Utilisez la commande `oc adm new-project` pour ajouter le sélecteur de nœud lors de la création du projet. Par exemple, la commande suivante crée un nouveau projet nommé `demo`, dans lequel tous les pods seront déployés sur les nœuds dont le libellé est `tier=1`.

```
[user@host ~]$ oc adm new-project demo --node-selector "tier=1"
```

Afin de configurer un sélecteur de nœud par défaut pour un projet existant, ajoutez une annotation à la ressource d'espace de noms avec la clé `openshift.io/node-selector`. La commande `oc annotate` permet d'ajouter, de modifier ou de supprimer une annotation de sélecteur de nœud :

```
[user@host ~]$ oc annotate namespace demo \
>   openshift.io/node-selector="tier=2" --overwrite
```

## Mise à l'échelle du nombre de répliques de pod

Bien que la plupart des ressources de déploiement commencent par la création d'un seul pod, le nombre de répliques (ou copies) d'un pod est souvent augmenté. Pour ce faire, il faut mettre à l'échelle le déploiement. Plusieurs méthodes de mise à l'échelle seront traitées ultérieurement, mais l'une de ces méthodes utilise la commande `oc scale`. Par exemple, le nombre de pods dans le déploiement `myapp` peut être porté à trois en utilisant la commande suivante :

```
[user@host ~]$ oc scale --replicas 3 deployment/myapp
```



### Références

Pour plus d'informations, reportez-vous au chapitre *Controlling pod placement onto nodes (scheduling)* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#controlling-pod-placement-onto-nodes-scheduling](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#controlling-pod-placement-onto-nodes-scheduling)

### Régions et zones de disponibilité d'Amazon Web Services

[https://aws.amazon.com/about-aws/global-infrastructure/regions\\_az/](https://aws.amazon.com/about-aws/global-infrastructure/regions_az/)

## ► Exercice guidé

# Contrôle du comportement de planification des pods

Dans cet exercice, vous allez configurer une application pour qu'elle s'exécute sur un sous-ensemble de nœuds de calcul du cluster.

## Résultats

Vous serez en mesure d'utiliser l'interface de ligne de commande OpenShift pour :

- Ajouter un nouveau libellé à un nœud.
- Déployer des pods sur les nœuds correspondant à une étiquette spécifique.
- Supprimer un libellé d'un nœud.
- Résoudre les problèmes en cas d'échec du déploiement des pods sur un nœud.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande permet de s'assurer que l'API du cluster est accessible et crée un projet que vous utiliserez au cours de l'activité.

```
[student@workstation ~]$ lab schedule-pods start
```

## Instructions

- 1. En tant qu'utilisateur `developer`, créez un nouveau projet nommé `schedule-pods`.

- 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Créez un nouveau projet nommé `schedule-pods`.

```
[student@workstation ~]$ oc new-project schedule-pods
Now using project "schedule-pods" on server "https://api.ocp4.example.com".

...output omitted...
```

- 2. Déployez et mettez à l'échelle une application de test.

**chapitre 13 |** Contrôle de la planification des pods

- 2.1. Créez une application nommée `hello` à l'aide du conteneur qui se trouve dans `quay.io/redhattraining/hello-world-nginx:v1.0`.

```
[student@workstation ~]$ oc new-app --name hello \
>   --image quay.io/redhattraining/hello-world-nginx:v1.0
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "hello" created
  deployment.apps "hello" created
  service "hello" created
--> Success
...output omitted...
```

- 2.2. Créez une route pour l'application.

```
[student@workstation ~]$ oc expose svc/hello
route.route.openshift.io/hello exposed
```

- 2.3. Mettez l'application à l'échelle manuellement pour qu'il y ait quatre pods en cours d'exécution.

```
[student@workstation ~]$ oc scale --replicas 4 deployment/hello
deployment.apps/hello scaled
```

- 2.4. Vérifiez que les quatre pods en cours d'exécution sont répartis entre les nœuds.

NAME	READY	STATUS	...	IP	NODE	...
hello-6c4984d949-78qsp	1/1	Running	...	10.9.0.30	master02	...
hello-6c4984d949-cf6tb	1/1	Running	...	10.10.0.20	master01	...
hello-6c4984d949-kwgbg	1/1	Running	...	10.8.0.38	master03	...
hello-6c4984d949-mb8z7	1/1	Running	...	10.10.0.19	master01	...

**Note**

En fonction de la charge existante sur chaque nœud, votre sortie peut être différente. Bien que le planificateur tente de répartir les pods, la distribution peut ne pas être uniforme.

- 3. Préparez les nœuds de façon à ce que les charges de travail applicatives puissent être distribuées à des nœuds de développement ou de production en affectant l'étiquette `env`. Affectez l'étiquette `env=dev` au nœud `master01` et l'étiquette `env=prod` au nœud `master02`.

- 3.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`. Un utilisateur normal n'est pas autorisé à afficher des informations sur les nœuds et ne peut pas étiqueter les nœuds.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 3.2. Vérifiez qu'aucun des nœuds n'utilise l'étiquette env.

```
[student@workstation ~]$ oc get nodes -L env
NAME      STATUS    ROLES     AGE      VERSION      ENV
master01   Ready     master,worker  42d     v1.23.3+e419edf
master02   Ready     master,worker  42d     v1.23.3+e419edf
master03   Ready     master,worker  42d     v1.23.3+e419edf
```

- 3.3. Ajoutez l'étiquette env=dev au nœud master01 pour indiquer qu'il s'agit d'un nœud de développement.

```
[student@workstation ~]$ oc label node master01 env=dev
node/master01 labeled
```

- 3.4. Ajoutez l'étiquette env=prod au nœud master02 pour indiquer qu'il s'agit d'un nœud de production.

```
[student@workstation ~]$ oc label node master02 env=prod
node/master02 labeled
```

- 3.5. Vérifiez que les nœuds ont le bon ensemble d'étiquettes env. Notez le nœud qui contient l'étiquette env=dev, car vous allez vérifier ultérieurement si les pods d'application ont été déployés sur ce nœud.

```
[student@workstation ~]$ oc get nodes -L env
NAME      STATUS    ROLES     AGE      VERSION      ENV
master01   Ready     master,worker  42d     v1.23.3+e419edf  dev
master02   Ready     master,worker  42d     v1.23.3+e419edf  prod
master03   Ready     master,worker  42d     v1.23.3+e419edf
```

- 4. Revenez à l'utilisateur developer et modifiez l'application hello de sorte qu'elle soit déployée sur le nœud de développement. Après avoir vérifié cette modification, supprimez le projet schedule-pods.

- 4.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur developer.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.

...output omitted...

Using project "schedule-pods".
```

**chapitre 13 |** Contrôle de la planification des pods

- 4.2. Modifiez la ressource `deployment` pour l'application `hello` afin de sélectionner un nœud de développement. Assurez-vous d'ajouter le sélecteur de nœuds au groupe `spec` dans la section `template`.

```
[student@workstation ~]$ oc edit deployment/hello
```

Ajoutez les lignes mises en surbrillance ci-dessous à la ressource de déploiement, en la mettant en retrait comme indiqué.

```
...output omitted...
  terminationMessagePath: /dev/termination-log
  terminationMessagePolicy: File
  dnsPolicy: ClusterFirst
  nodeSelector:
    env: dev
  restartPolicy: Always
...output omitted...
```

La sortie suivante de `oc edit` s'affiche une fois que vous avez enregistré vos modifications.

```
deployment.apps/hello edited
```

- 4.3. Vérifiez que les pods d'application sont déployés sur le nœud doté de l'étiquette `env=dev`. Bien que le redéploiement puisse prendre du temps, les pods d'application doivent être déployés sur le nœud `master01`.

```
[student@workstation ~]$ oc get pods -o wide
NAME           READY   STATUS    RESTARTS   AGE     IP          NODE      ...
hello-b556ccf8b-8scxd  1/1    Running   0          80s   10.10.0.14  master01  ...
hello-b556ccf8b-hb24w  1/1    Running   0          77s   10.10.0.16  master01  ...
hello-b556ccf8b-qxlj8  1/1    Running   0          80s   10.10.0.15  master01  ...
hello-b556ccf8b-sdpxpj 1/1    Running   0          76s   10.10.0.17  master01  ...
```

- 4.4. Supprimez le projet `schedule-pods`.

```
[student@workstation ~]$ oc delete project schedule-pods
project.project.openshift.io "schedule-pods" deleted
```

- 5. Terminez le nettoyage de cette partie de l'exercice en supprimant l'étiquette `env` de tous les nœuds.

- 5.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 5.2. Supprimez l'étiquette `env` de tous les nœuds qui l'ont.

```
[student@workstation ~]$ oc label node -l env env-
node/master01 labeled
node/master02 labeled
```

- 6. Le projet `schedule-pods-ts` contient une application qui s'exécute uniquement sur les nœuds étiquetés `client=ACME`. Dans l'exemple suivant, le pod d'application est en attente et vous devez diagnostiquer le problème à l'aide des étapes suivantes :

- 6.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `developer` et assurez-vous que vous utilisez le projet `schedule-pods-ts`.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.

...output omitted...

Using project "schedule-pods-ts".
```

Si le résultat ci-dessus ne montre pas que vous utilisez le projet `schedule-pods-ts`, basculez vers celui-ci.

```
[student@workstation ~]$ oc project schedule-pods-ts
Now using project "schedule-pods-ts" on server ...
```

- 6.2. Vérifiez que l'état du pod d'application est `Pending`.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-ts-5dbff9f44-w6csj   0/1     Pending   0          6m19s
```

- 6.3. Étant donné qu'un pod dont l'état est `Pending` ne dispose d'aucun journal, vérifiez les détails du pod à l'aide de la commande `oc describe pod` pour voir si la description du pod fournit des informations utiles.

```
[student@workstation ~]$ oc describe pod hello-ts-5dbff9f44-8h7c7
...output omitted...
QoS Class:      BestEffort
Node-Selectors:  client=acme
Tolerations:    node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type      Reason            ...
  ----      -----            ...
  Warning  FailedScheduling  ...
  ...
  Warning  FailedScheduling  ...  0/3 nodes are available: 3 node(s) didn't match
                                node selector.
```

À partir de ces informations, le pod doit être planifié sur un nœud doté de l'étiquette `client=acme`, mais aucun des trois nœuds n'a cette étiquette.

- 6.4. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin` et vérifiez le libellé du nœud de calcul. Pour le vérifier, exécutez `oc get nodes -L client` afin de lister les détails des nœuds disponibles.

```
[student@workstation ~]$ oc login -u admin -p redhat
```

```
Login successful.
```

```
...output omitted...
```

```
[student@workstation ~]$ oc get nodes -L client
```

NAME	STATUS	ROLES	AGE	VERSION	CLIENT
master01	Ready	master,worker	10d	v1.23.3+e419edf	ACME

Les informations fournies indiquent qu'au moins un nœud de calcul a l'étiquette **client=ACME**. Vous avez trouvé le problème. L'application doit être modifiée de sorte qu'elle utilise le bon sélecteur de nœud.

- 6.5. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur developer et modifiez la ressource de déploiement pour que l'application utilise le bon sélecteur de nœud.

```
[student@workstation ~]$ oc login -u developer -p developer
```

```
Login successful.
```

```
...output omitted...
```

```
[student@workstation ~]$ oc edit deployment/hello-ts
```

Remplacez acme par ACME, comme illustré ci-dessous.

```
...output omitted...
dnsPolicy: ClusterFirst
nodeSelector:
  client: ACME
restartPolicy: Always
...output omitted...
```

La sortie suivante de `oc edit` s'affiche une fois que vous avez enregistré vos modifications.

```
deployment.apps/hello-ts edited
```

- 6.6. Vérifiez qu'un nouveau pod d'application est déployé et que son état est Running.

```
[student@workstation ~]$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
hello-ts-69769f64b4-wwhpc	1/1	Running	0	11s

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab schedule-pods finish
```

L'exercice guidé est maintenant terminé.

# Limitation de l'utilisation des ressources par une application

---

## Résultats

Après avoir terminé cette section, vous devez pouvoir limiter les ressources utilisées par des conteneurs, des pods et des projets.

## Définition des demandes et limites de ressources pour les pods

Une définition de pod peut inclure à la fois des demandes de ressources et des limites de ressources :

### Demandes de ressources

Elles sont utilisées pour la planification et pour indiquer qu'un pod ne peut pas s'exécuter avec une quantité de ressources de calcul inférieure à celle qui est spécifiée. Le planificateur essaie de trouver un nœud ayant des ressources de calcul suffisantes pour satisfaire les demandes du pod.

### Limites de ressources

Elles sont utilisées pour empêcher qu'un pod n'utilise toutes les ressources informatiques d'un nœud. Le nœud qui exécute un pod configure la fonction cgroups du noyau Linux afin d'appliquer les limites de ressources du pod.

Les demandes de ressources et les limites de ressources doivent être définies pour chaque conteneur dans une ressource de déploiement ou de configuration de déploiement. Si les demandes et les limites n'ont pas été définies, vous trouverez une ligne resources: {} pour chaque conteneur.

Modifiez la ligne resources: {} pour spécifier les demandes ou les limites souhaitées. Par exemple :

```
...output omitted...
spec:
  containers:
    - image: quay.io/redhattraining/hello-world-nginx:v1.0
      name: hello-world-nginx
      resources:
        requests:
          cpu: "10m"
          memory: 20Mi
        limits:
          cpu: "80m"
          memory: 100Mi
  status: {}
```

Si vous utilisez la commande oc edit pour modifier un déploiement ou une configuration de déploiement, assurez-vous d'utiliser la bonne mise en retrait. Les erreurs de mise en retrait peuvent entraîner le refus de l'éditeur d'enregistrer les modifications. Pour éviter les problèmes de mise en retrait, vous pouvez utiliser la commande oc set resources pour spécifier les

demandes et limites de ressources. La commande suivante définit les mêmes demandes et limites que dans l'exemple précédent :

```
[user@host ~]$ oc set resources deployment hello-world-nginx \
>   --requests cpu=10m,memory=20Mi --limits cpu=80m,memory=100Mi
```

Si un quota de ressources s'applique à une demande de ressources, le pod doit alors définir une demande de ressources. Si un quota de ressources s'applique à une limite de ressources, le pod doit alors aussi définir une limite de ressources. Red Hat recommande de définir des demandes et des limites de ressources, même si les quotas ne sont pas utilisés.

## Affichage des demandes, des limites et de l'utilisation réelle

À l'aide de l'interface de ligne de commande OpenShift, les administrateurs de cluster peuvent afficher des informations sur l'utilisation des calculs sur des nœuds individuels. La commande `oc describe node` affiche des informations détaillées sur un nœud, notamment des informations sur les pods en cours d'exécution sur le nœud. Pour chaque pod, elle affiche les demandes et les limites processeur, ainsi que les demandes et les limites de mémoire. Si une demande ou une limite n'a pas été spécifiée, le pod affichera alors un 0 pour cette colonne. Un résumé de toutes les demandes et limites de ressources est également affiché.

```
[user@host ~]$ oc describe node node1.us-west-1.compute.internal
Name:           node1.us-west-1.compute.internal
Roles:          worker
Labels:         beta.kubernetes.io/arch=amd64
                beta.kubernetes.io/instance-type=m4.xlarge
                beta.kubernetes.io/os=linux
...
Non-terminated Pods:             (20 in total)
...  Name                  CPU Requests  ...  Memory Requests  Memory Limits  AGE
...  ----                  -----        ...  -----          -----          -----
...  tuned-vdwt4            10m (0%)    ...  50Mi (0%)      0 (0%)       8d
...  dns-default-2rpwf     110m (3%)   ...  70Mi (0%)      512Mi (3%)    8d
...  node-ca-6xwmn         10m (0%)    ...  10Mi (0%)      0 (0%)       8d
...
Resource          Requests      Limits
-----          -----
cpu              600m (17%)   0 (0%)
memory          1506Mi (9%)  512Mi (3%)
...

```



### Note

Les colonnes de synthèse pour `Requests` et `Limits` affichent les totaux des demandes et limites définies. Dans la sortie ci-dessus, un seul des 20 pods en cours d'exécution sur le nœud définissait une limite de mémoire, et cette limite était de 512 Mi.

La commande `oc describe node` affiche les demandes et les limites, tandis que la commande `oc adm top` affiche l'utilisation réelle. Par exemple, si un pod demande 10 m de processeur, le planificateur s'assure qu'il place le pod sur un nœud doté de suffisamment de capacité disponible.

Bien que le pod ait demandé 10 m de processeur, il peut utiliser plus ou moins de cette valeur, à moins qu'il ne soit également contraint par une limite processeur. De la même façon, un pod qui ne spécifie pas de demandes de ressources utilisera toujours une certaine quantité de ressources. La commande `oc adm top nodes` affiche l'utilisation réelle d'un ou de plusieurs nœuds du cluster, tandis que la commande `oc adm top pods` affiche l'utilisation réelle de chaque pod d'un projet.

[user@host ~]\$ oc adm top nodes -l node-role.kubernetes.io/worker				
NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
node1.us-west-1.compute.internal	519m	14%	3126Mi	20%
node2.us-west-1.compute.internal	167m	4%	1178Mi	7%

## Application de quotas

OpenShift Container Platform peut appliquer des quotas qui surveillent et limitent l'utilisation de deux types de ressources :

### Nombres d'objets

Le nombre de ressources Kubernetes, comme les pods, les services et les routages.

### Ressources informatiques

Quantité de ressources physiques ou virtuelles, comme le processeur, la mémoire et la capacité de stockage.

Le fait d'imposer un quota sur la quantité de ressources Kubernetes améliore la stabilité du plan de contrôle OpenShift en évitant une croissance illimitée de la base de données Etcd. L'instauration de quotas sur les ressources Kubernetes évite également d'épuiser d'autres ressources logicielles limitées, comme les adresses IP pour les services.

De la même façon, le fait d'imposer un quota sur la quantité de ressources de calcul évite d'épuiser la capacité de calcul d'un seul nœud dans un cluster OpenShift. Cela évite également qu'une application prive les autres applications qui partagent le cluster en utilisant toutes les capacités du cluster.

OpenShift gère les quotas pour le nombre de ressources et l'utilisation des ressources de calcul dans un cluster en utilisant une ressource `ResourceQuota`, ou un `quota`. Un quota spécifie des limites strictes d'utilisation des ressources pour un projet. Tous les attributs d'un quota sont facultatifs, ce qui signifie que toute ressource non limitée par un quota peut être utilisée sans limite.



#### Note

Bien qu'une seule ressource de quota puisse définir tous les quotas d'un projet, un projet peut également contenir plusieurs quotas. Par exemple, une ressource de quota peut limiter les ressources de calcul, telles que la mémoire totale ou le nombre total de processeurs autorisés. Une autre ressource de quota peut limiter les comptes d'objets, tels que le nombre de pods autorisés ou le nombre de services autorisés.

L'effet de plusieurs quotas est cumulatif, mais l'on s'attend à ce que deux ressources `ResourceQuota` différentes ne limitent pas l'utilisation du même type de ressources Kubernetes ou de ressources de calcul. Par exemple, deux quotas différents dans un projet ne doivent pas tous deux essayer de limiter le nombre maximal de pods autorisés.

Le tableau suivant décrit certaines ressources qu'un quota peut limiter en fonction de leur nombre ou numéro :

Nom de la ressource	Description du quota
pods	Nombre total de pods
replicationcontrollers	Nombre total de contrôleurs de réPLICATION
services	Nombre total de services
secrets	Nombre total de secrets
persistenTvolumeclaims	Nombre total de revendications de volume persistant

Le tableau suivant décrit certaines ressources de calcul pouvant être limitées par un quota :

Nom de la ressource de calcul	Description du quota
requests.cpu	Utilisation totale de processeur sur tous les conteneurs
requests.memory	Utilisation totale de mémoire sur tous les conteneurs
requests.storage	Nombre total de demandes de stockage par conteneurs parmi toutes les revendications de volume persistant

Les attributs de quota peuvent surveiller les demandes de ressources ou les limites de ressources pour tous les pods du projet. Par défaut, les attributs de quota surveillent les demandes de ressources. Pour surveiller les limites de ressources, ajoutez plutôt le préfixe `limits` au nom de la ressource de calcul ; `limits.cpu`, par exemple.

La liste suivante montre une ressource `ResourceQuota` à l'aide de la syntaxe YAML. Cet exemple spécifie les quotas pour le nombre de ressources et l'utilisation des ressources de calcul :

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
spec:
  hard:
    services: "10"
    cpu: "1300m"
    memory: "1.5Gi"
```

Les unités de ressources sont les mêmes pour les limites de ressources et les demandes de ressources des pods. Par exemple : `Gi` signifie Gio (gibioctet) et `m` signifie millicores. Un millicore est l'équivalent de 1/1 000 d'un seul cœur de processeur.

Les quotas de ressources peuvent être créés de la même façon que n'importe quelle autre ressource OpenShift Container Platform ; c'est-à-dire en transmettant un fichier de définitions de ressources JSON ou YAML à la commande `oc create` :

```
[user@host ~]$ oc create --save-config -f dev-quota.yml
```

Un autre moyen de créer un quota de ressource consiste à utiliser la commande `oc create quota`, par exemple :

```
[user@host ~]$ oc create quota dev-quota --hard services=10,cpu=1300,memory=1.5Gi
```

Utilisez la commande `oc get resourcequota` pour dresser la liste des quotas disponibles et utilisez la commande `oc describe resourcequota` pour consulter les statistiques d'utilisation liées à toute limite stricte définie dans le quota, par exemple :

```
[user@host ~]$ oc get resourcequota
NAME          AGE   REQUEST
compute-quota 51s   cpu: 500m/10, memory: 300Mi/1Gi    ...
count-quota   28s   pods: 1/3, replicationcontrollers: 1/5, services: 1/2 ...
```

Sans arguments, la commande `oc describe quota` affiche les limites cumulées définies pour toutes les ressources ResourceQuota du projet :

```
[user@host ~]$ oc describe quota
Name:           compute-quota
Namespace:      schedule-demo
Resource        Used   Hard
-----  -----
cpu            500m   10
memory         300Mi  1Gi

Name:           count-quota
Namespace:      schedule-demo
Resource        Used   Hard
-----  -----
pods           1       3
replicationcontrollers 1       5
services        1       2
```

Un quota actif peut être supprimé par nom à l'aide de la commande `oc delete` :

```
[user@host ~]$ oc delete resourcequota QUOTA
```

Quand un quota est créé pour la première fois dans un projet, ce dernier restreint la capacité à créer toute nouvelle ressource susceptible d'enfreindre une contrainte de quota jusqu'à ce qu'il ait calculé les statistiques d'utilisation mises à jour. Une fois qu'un quota a été créé et que les statistiques d'utilisation sont à jour, le projet accepte la création de contenu.

Lorsqu'une ressource est créée, l'utilisation du quota est immédiatement incrémentée. Lorsqu'une ressource est supprimée, l'utilisation du quota est décrémentée au cours du recalcul complet suivant des statistiques de quota pour le projet.

Les quotas sont appliqués aux nouvelles ressources, mais ils n'affectent pas les ressources existantes. Par exemple, si vous créez un quota pour limiter un projet à 15 pods, mais que 20 pods sont déjà en cours d'exécution, le quota ne supprimera alors pas les 5 pods supplémentaires qui dépassent le quota.

**Important**

Des contraintes ResourceQuota sont appliquées au projet dans son ensemble, mais un grand nombre de processus OpenShift, comme les compilations et les déploiements, créent des pods à l'intérieur du projet et peuvent échouer, car leur démarrage entraînerait un dépassement du quota du projet.

Si la modification d'un projet entraîne un dépassement du quota pour un nombre de ressources, OpenShift refuse l'action et renvoie un message d'erreur approprié à l'utilisateur. Cependant, si la modification dépasse le quota pour une ressource de calcul, l'opération n'échoue pas immédiatement ; OpenShift retente l'opération plusieurs fois, en offrant à l'administrateur l'occasion d'augmenter le quota ou d'effectuer une autre opération corrective, comme la mise en ligne d'un nouveau nœud.

**Important**

Si un quota limitant l'utilisation des ressources de calcul pour un projet est défini, OpenShift refuse alors de créer des modules ne spécifiant pas de demandes de ressources ni de limites de ressources pour cette ressource de calcul. Pour utiliser la plupart des modèles et outils de création, avec un projet restreint par des quotas, le projet doit également contenir une ressource de plage de limites qui spécifie des valeurs par défaut pour les demandes de ressources de conteneur.

## Application de plages de limites

Une ressource `LimitRange`, également appelée `limit`, définit les valeurs par défaut, minimales et maximales pour les demandes de ressources de calcul, ainsi que les limites pour un seul pod ou conteneur défini dans le projet. La demande ou la limite de ressources pour un pod est la somme de ses conteneurs.

Pour comprendre la différence entre une plage de limites et un quota de ressources, considérez qu'une plage de limites définit des plages et des valeurs par défaut valides pour un seul pod, et un quota de ressources définit uniquement des valeurs maximales pour la somme de tous les pods d'un projet. Un administrateur de cluster soucieux de l'utilisation des ressources dans un cluster OpenShift définit généralement à la fois des limites et des quotas pour un projet.

Une ressource de plage de limite peut également définir des valeurs par défaut, minimales et maximales pour la capacité de stockage demandée par une image, un flux d'images ou une revendication de volume persistant. Si une ressource ajoutée à un projet ne fournit pas de demande de ressources de calcul, elle prend alors la valeur par défaut fournie par les plages de limites pour le projet.

Si une nouvelle ressource fournit des demandes ou des limites de ressources de calcul inférieures au minimum spécifié par les plages de limites du projet, la ressource n'est pas créée. De même, si une nouvelle ressource fournit des demandes ou des limites de ressources de calcul supérieures au maximum spécifié par les plages de limites du projet, la ressource n'est pas créée.

La liste suivante montre une plage de limites définie à l'aide de la syntaxe YAML :

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "dev-limits"
```

```
spec:  
  limits:  
    - type: "Pod"  
      max: ①  
        cpu: "500m"  
        memory: "750Mi"  
      min: ②  
        cpu: "10m"  
        memory: "5Mi"  
    - type: "Container"  
      max: ③  
        cpu: "500m"  
        memory: "750Mi"  
      min: ④  
        cpu: "10m"  
        memory: "5Mi"  
      default: ⑤  
        cpu: "100m"  
        memory: "100Mi"  
      defaultRequest: ⑥  
        cpu: "20m"  
        memory: "20Mi"  
    - type: openshift.io/Image ⑦  
      max:  
        storage: 1Gi  
    - type: openshift.io/ImageStream ⑧  
      max:  
        openshift.io/image-tags: 10  
        openshift.io/images: 20  
    - type: "PersistentVolumeClaim" ⑨  
      min:  
        storage: "1Gi"  
      max:  
        storage: "50Gi"
```

- ① Quantité maximale de processeurs et de mémoire que tous les conteneurs d'un pod peuvent utiliser. Un pod qui dépasse les limites maximales n'est pas créé. Un pod existant qui dépasse les limites maximales est redémarré.
- ② Quantité minimale de processeurs et de mémoire utilisée entre tous les conteneurs d'un pod. Un pod qui ne satisfait pas à la configuration minimale requise n'est pas créé. Étant donné que de nombreux pods ne comportent qu'un seul conteneur, vous pouvez définir les valeurs de pod minimales sur les mêmes valeurs que les valeurs de conteneur minimales.
- ③ Quantité maximale de processeurs et de mémoire que chaque conteneur d'un pod peut utiliser. Un nouveau conteneur qui dépasse les limites maximales ne crée pas le pod associé. Un conteneur existant qui dépasse les limites maximales redémarre l'ensemble du pod.
- ④ Quantité minimale de processeurs et de mémoire que chaque conteneur d'un pod peut utiliser. Un conteneur qui ne répond pas à la configuration minimale requise empêche la création du pod associé.
- ⑤ Quantité maximale par défaut de processeurs et de mémoire qu'un conteneur peut utiliser. Ce paramètre est utilisé si aucune limite de ressources processeur ou limite de mémoire n'est définie pour le conteneur.

- ⑥ Valeurs de processeur et de mémoire par défaut demandées par un conteneur. Ce paramètre par défaut est utilisé si aucune demande de ressources processeur ou demande de mémoire n'est définie pour le conteneur. Si les quotas de processeur et de mémoire sont activés pour un espace de noms, la configuration de la section `defaultRequest` permet aux pods de démarrer, même si les conteneurs ne spécifient pas de demandes de ressources.
- ⑦ Taille maximale d'une image qui peut être chargée sur le registre interne
- ⑧ Nombre maximal de versions et de balises d'image auquel une ressource de flux d'images peut faire référence.
- ⑨ Tailles minimale et maximale autorisées pour une revendication de volume persistant.

Les utilisateurs peuvent créer une ressource de plage limite de la même façon que n'importe quelle autre ressource OpenShift ; c'est-à-dire en transmettant un fichier de définitions de ressources JSON ou YAML à la commande `oc create` :

```
[user@host ~]$ oc create --save-config -f dev-limits.yml
```

Red Hat OpenShift Container Platform ne fournit pas une commande `oc create` spécifiquement pour les plages de limites comme c'est le cas pour les quotas de ressources. La seule alternative est d'utiliser des fichiers YAML ou JSON.

Utilisez la commande `oc describe limitrange` pour consulter les contraintes de limites appliquées dans un projet :

```
[user@host ~]$ oc describe limitrange dev-limits
Name:          dev-limits
Namespace:    schedule-demo
Type           Resource     Min   Max   Default Request ...
-----        -----      ---   ---   -----  -----
Pod           cpu         10m   500m  -       ...
Pod           memory      5Mi   750Mi -       ...
Container     memory      5Mi   750Mi 20Mi  ...
Container     cpu         10m   500m  20m  ...
openshift.io/Image storage   -     1Gi   -       ...
openshift.io/ImageStream openshift.io/image-tags -     10    -       ...
openshift.io/ImageStream openshift.io/images    -     20    -       ...
PersistentVolumeClaim storage  1Gi   50Gi  -       ...
```

Une plage de limites active peut être supprimée à l'aide de la commande `oc delete` :

```
[user@host ~]$ oc delete limitrange dev-limits
```

Après qu'une plage de limites a été créée dans un projet, toutes les demandes de création de nouvelles ressources sont évaluées en fonction de chaque ressource de plage de limite du projet. Si la nouvelle ressource enfreint la contrainte de minimum ou de maximum indiquée par une plage de limite, la ressource est refusée. Si la nouvelle ressource ne définit pas de valeur explicite, et que la contrainte prend en charge une valeur par défaut, cette dernière est appliquée à la nouvelle ressource comme valeur d'utilisation.

Toutes les demandes de mise à jour de ressources sont également évaluées en fonction de chaque ressource de plage de limites du projet. Si la ressource mise à jour enfreint une contrainte, la mise à jour est refusée.

**Important**

Évitez de définir des contraintes `LimitRange` trop élevées, ou des contraintes `ResourceQuota` trop basses. Le non-respect de contraintes `LimitRange` empêche la création du pod, ce qui entraîne l'affichage de messages d'erreur. Le non-respect de contraintes `ResourceQuota` empêche la planification d'un pod sur un nœud. Le pod peut être créé mais reste à l'état Pending.

## Application de quotas à plusieurs projets

La ressource `ClusterResourceQuota` est créée au niveau du cluster, de la même façon que pour un volume persistant, et spécifie des contraintes de ressources qui s'appliquent à plusieurs projets.

Les administrateurs de cluster peuvent spécifier les projets sujets à des quotas de ressources de cluster de deux façons :

- En utilisant une annotation `openshift.io/requester` pour spécifier le propriétaire du projet. Tous les projets dont le propriétaire est spécifié sont assujettis au quota.
- En utilisant un sélecteur. Tous les projets dont les libellés correspondent au sélecteur sont assujettis au quota.

Ci-dessous, un exemple de création d'un quota de ressources de cluster pour tous les projets appartenant à l'utilisateur qa :

```
[user@host ~]$ oc create clusterquota user-qa \
>   --project-annotation-selector openshift.io/requester=qa \
>   --hard pods=12,secrets=20
```

Ci-dessous, un exemple de création d'un quota de ressources de cluster pour tous les projets qui ont été affectés à l'étiquette `environment=qa` :

```
[user@host ~]$ oc create clusterquota env-qa \
>   --project-label-selector environment=qa \
>   --hard pods=10,services=5
```

Les utilisateurs d'un projet peuvent utiliser la commande `oc describe QUOTA` pour consulter les quotas de ressources de cluster qui s'appliquent au projet en cours, le cas échéant.

Utilisez la commande `oc delete` pour supprimer un quota de ressources de cluster :

```
[user@host ~]$ oc delete clusterquota QUOTA
```

**Note**

Pour éviter d'importantes charges de verrouillage, il est déconseillé d'avoir un seul quota de ressources correspondant à plus d'une centaine de projets. Lors de la mise à jour ou de la création des ressources d'un projet, le projet est verrouillé le temps de rechercher les quotas de ressources applicables.

## Personnalisation du modèle de projet par défaut

Les administrateurs du cluster peuvent personnaliser le modèle de projet par défaut. Des ressources supplémentaires, telles que des quotas, des plages de limites et des politiques réseau, sont créées lorsqu'un utilisateur crée un projet.

En tant qu'administrateur du cluster, créez un modèle de projet à l'aide de la commande `oc adm create-bootstrap-project-template`, puis redirigez la sortie vers un fichier :

```
[user@host ~]$ oc adm create-bootstrap-project-template \
> -o yaml > /tmp/project-template.yaml
```

Personnalisez le fichier de ressources du modèle pour ajouter des ressources supplémentaires, telles que des quotas, des plages de limites et des politiques réseau. Pour rappel, les quotas sont configurés par les administrateurs du cluster et ne peuvent pas être ajoutés, modifiés ni supprimés par des administrateurs de projet. Les administrateurs de projet peuvent modifier et supprimer des plages de limites et des politiques réseau, même si ces ressources sont créées par le modèle de projet.

Les nouveaux projets créent les ressources spécifiées dans la section `objects`. Les objets supplémentaires doivent respecter la même mise en retrait que les ressources `Project` et `RoleBinding`.

L'exemple suivant crée un quota à l'aide du nom du projet ; le quota impose une limite de 3 processeurs, 10 Go de mémoire et 10 pods sur le projet :

```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
  creationTimestamp: null
  name: project-request
objects:
- apiVersion: project.openshift.io/v1
  kind: Project
  ...output omitted...
- apiVersion: rbac.authorization.k8s.io/v1
  kind: RoleBinding
  ...output omitted...
- apiVersion: v1
  kind: ResourceQuota
  metadata:
    name: ${PROJECT_NAME}-quota
  spec:
    hard:
      cpu: "3"
      memory: 10Gi
      pods: "10"
parameters:
- name: PROJECT_NAME
- name: PROJECT_DISPLAYNAME
- name: PROJECT_DESCRIPTION
- name: PROJECT_ADMIN_USER
- name: PROJECT_REQUESTING_USER
```

Utilisez la commande `oc create` pour créer une ressource de modèle dans l'espace de noms `openshift-config`:

```
[user@host ~]$ oc create -f project-template.yaml -n openshift-config  
template.template.openshift.io/project-request created
```

Mettez à jour la ressource `projects.config.openshift.io/cluster` pour utiliser le nouveau modèle de projet. Modifiez la section `spec`. Par défaut, le nom du modèle de projet est `project-request`.

```
apiVersion: config.openshift.io/v1  
kind: Project  
metadata:  
...output omitted...  
  name: cluster  
...output omitted...  
spec:  
  projectRequestTemplate:  
    name: project-request
```

En cas de mise à jour réussie de la ressource `projects.config.openshift.io/cluster`, de nouveaux pods `apiserver` sont créés dans l'espace de noms `openshift-apiserver`. Une fois que les nouveaux pods `apiserver` sont en cours d'exécution, de nouveaux projets créent les ressources spécifiées dans le modèle de projet personnalisé.

Pour revenir au modèle de projet d'origine, modifiez la ressource `projects.config.openshift.io/cluster` afin d'effacer la ressource `spec` de sorte qu'elle corresponde à : `spec: {}`



## Références

Pour plus d'informations, reportez-vous au chapitre *Quotas* de la documentation *Building applications* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse  
[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/building\\_applications/index#quotas](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/building_applications/index#quotas)

Pour plus d'informations sur les plages de limites, reportez-vous à la section *Restrict resource consumption with limit ranges* du chapitre *Working with clusters* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse  
[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#nodes-cluster-limit-ranges](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#nodes-cluster-limit-ranges)

### Personnalisation de la création d'un projet OpenShift

<https://developers.redhat.com/blog/2020/02/05/customizing-openshift-project-creation/>

## ► Exercice guidé

# Limitation de l'utilisation des ressources par une application

Dans cet exercice, vous allez configurer une application de sorte qu'elle ne consomme pas toutes les ressources de calcul du cluster et de ses nœuds de calcul.

## Résultats

Vous serez en mesure d'utiliser l'interface de ligne de commande OpenShift pour :

- Configurer une application pour spécifier les demandes de ressources pour l'utilisation du processeur et de la mémoire.
- Modifier une application pour qu'elle puisse fonctionner dans les restrictions de cluster existantes.
- Créer un quota pour limiter la quantité totale de processeur, de mémoire et de mappages de configuration disponibles pour un projet.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande permet de s'assurer que l'API du cluster est accessible et crée les fichiers de ressources que vous utiliserez au cours de l'activité.

```
[student@workstation ~]$ lab schedule-limit start
```

## Instructions

- 1. En tant qu'utilisateur `developer`, créez un nouveau projet nommé `schedule-limit`.
- 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Créez un nouveau projet pour cet exercice guidé nommé `schedule-limit`.

```
[student@workstation ~]$ oc new-project schedule-limit
Now using project "schedule-limit" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 2. Déployez une application de test pour cet exercice qui demande explicitement des ressources de conteneur pour le processeur et la mémoire.
- 2.1. Créez un fichier de ressources de déploiement et enregistrez-le dans ~/D0280/labs/schedule-limit/hello-limit.yaml. Nommez l'application hello-limit et utilisez l'image de conteneur située sous quay.io/redhattraining/hello-world-nginx:v1.0.

```
[student@workstation ~]$ oc create deployment hello-limit \
>   --image quay.io/redhattraining/hello-world-nginx:v1.0 \
>   --dry-run=client -o yaml > ~/D0280/labs/schedule-limit/hello-limit.yaml
```

- 2.2. Modifiez ~/D0280/labs/schedule-limit/hello-limit.yaml pour remplacer la ligne resources: {} par les lignes mises en surbrillance ci-dessous.

```
[student@workstation ~]$ vim ~/D0280/labs/schedule-limit/hello-limit.yaml
```

Assurez-vous que la mise en retrait est bonne avant d'enregistrer le fichier.

```
...output omitted...
spec:
  containers:
    - image: quay.io/redhattraining/hello-world-nginx:v1.0
      name: hello-world-nginx
      resources:
        requests:
          cpu: "3"
          memory: 20Mi
  status: {}
```

- 2.3. Créez la nouvelle application à l'aide de votre fichier de ressources.

```
[student@workstation ~]$ oc create --save-config \
>   -f ~/D0280/labs/schedule-limit/hello-limit.yaml
deployment.apps/hello-limit created
```

- 2.4. Bien qu'un nouveau déploiement ait été créé pour l'application, le pod d'application doit avoir le statut Pending.

```
[student@workstation ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
hello-limit-d86874d86b-fpmrt   0/1     Pending   0          10s
```

- 2.5. Le pod ne peut pas être planifié, car aucun des nœuds de calcul ne dispose de suffisamment de ressources processeur. Cette opération peut être vérifiée en affichant les événements d'avertissement.

```
[student@workstation ~]$ oc get events --field-selector type=Warning
LAST SEEN   TYPE      REASON           OBJECT                   MESSAGE
88s         Warning   FailedScheduling   pod/hello-limit-d86874d86b-fpmrt  0/3
nodes are available: 3 Insufficient cpu.
```

- 3. Redéployez votre application de sorte qu'elle demande moins de ressources processeur.
- 3.1. Modifiez `~/D0280/labs/schedule-limit/hello-limit.yaml` pour demander 1 processeur pour le conteneur.

```
[student@workstation ~]$ vim ~/D0280/labs/schedule-limit/hello-limit.yaml
```

Modifiez la ligne `cpu: "3"` pour qu'elle corresponde à la ligne en surbrillance ci-dessous.

```
...output omitted...
resources:
  requests:
    cpu: "1200m"
    memory: 20Mi
```

- 3.2. Appliquez les modifications à votre application.

```
[student@workstation ~]$ oc apply -f ~/D0280/labs/schedule-limit/hello-limit.yaml
deployment.apps/hello-limit configured
```

- 3.3. Vérifiez que votre application a bien été déployée. Vous pouvez avoir à exécuter `oc get pods` plusieurs fois jusqu'à ce que vous rencontriez un pod en cours d'exécution. Le pod précédent en état Pending s'interrompra et finira par disparaître.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS      RESTARTS   AGE
hello-limit-d86874d86b-fpmrt   0/1     Terminating   0          2m19s
hello-limit-7c7998ff6b-ctsjp   1/1     Running      0          6s
```



#### Note

Si votre pod d'application n'est pas planifié, modifiez le fichier `~/D0280/labs/schedule-limit/hello-limit.yaml` pour réduire la demande de processeur à `1100m`. Appliquez à nouveau les modifications et vérifiez que le statut du pod est `Running`.

- 4. Essayez de mettre à l'échelle votre application d'un pod à quatre pods. Après avoir vérifié que cette modification dépasserait la capacité de votre cluster, supprimez les ressources associées à l'application `hello-limit`.

- 4.1. Mettez à l'échelle manuellement l'application `hello-limit` à quatre pods.

```
[student@workstation ~]$ oc scale --replicas 4 deployment/hello-limit
deployment.apps/hello-limit scaled
```

- 4.2. Vérifiez que les quatre pods sont en cours d'exécution. Vous pouvez avoir à exécuter `oc get pods` plusieurs fois jusqu'à ce qu'un pod au moins soit en attente. En fonction de la charge de cluster actuelle, plusieurs pods supplémentaires peuvent être en attente.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-limit-d55cd65c-765s9  1/1     Running   0          12s
hello-limit-d55cd65c-hmblw  0/1     Pending    0          12s
hello-limit-d55cd65c-r8lvw  1/1     Running   0          12s
hello-limit-d55cd65c-vkrhk  0/1     Pending    0          12s
```

**Note**

Si votre pod d'application ne se déploie toujours pas, définissez le nombre de pods d'application sur zéro, puis modifiez ~/D0280/labs/schedule-limit/hello-limit.yaml pour réduire la demande de processeur à 1000m.

Exécutez `oc apply -f ~/D0280/labs/schedule-limit/hello-limit.yaml` pour appliquer les modifications, puis exécutez à nouveau la commande `oc scale` pour faire passer le nombre de pods à quatre.

- 4.3. Les événements d'avertissement indiquent qu'un ou plusieurs pods ne peuvent pas être planifiés, car aucun des nœuds ne dispose de suffisamment de ressources processeur. Vos messages d'avertissement peuvent être légèrement différents.

```
[student@workstation ~]$ oc get events --field-selector type=Warning
LAST SEEN      TYPE        REASON               OBJECT
MESSAGE
...output omitted...
76s           Warning     FailedScheduling   pod/hello-limit-d55cd65c-vkrhk      0/3
nodes are available: 3 Insufficient cpu.
```

- 4.4. Supprimez toutes les ressources associées à l'application `hello-limit`.

```
[student@workstation ~]$ oc delete all -l app=hello-limit
pod "hello-limit-d55cd65c-765s9" deleted
pod "hello-limit-d55cd65c-hmblw" deleted
pod "hello-limit-d55cd65c-r8lvw" deleted
pod "hello-limit-d55cd65c-vkrhk" deleted
deployment.apps "hello-limit" deleted
replicaset.apps "hello-limit-5cc86ff6b8" deleted
replicaset.apps "hello-limit-7d6bdcc99b" deleted
```

- 5. Déployez une deuxième application pour tester l'utilisation de mémoire. Cette seconde application définit une limite de mémoire de 200 Mo par conteneur.

- 5.1. Utilisez le fichier de ressources situé à l'emplacement ~/D0280/labs/schedule-limit/loadtest.yaml pour créer l'application `loadtest`. En plus de créer un déploiement, ce fichier de ressources crée également un service et une route.

```
[student@workstation ~]$ oc create --save-config \
>   -f ~/D0280/labs/schedule-limit/loadtest.yaml
deployment.apps/loadtest created
service/loadtest created
route.route.openshift.io/loadtest created
```

- 5.2. L'image de conteneur `loadtest` est conçue pour augmenter la charge processeur ou mémoire sur le conteneur en effectuant une demande à l'API de l'application. Identifiez le nom de domaine complet utilisé dans la route.

```
[student@workstation ~]$ oc get routes
NAME      HOST/PORT
loadtest  loadtest.apps.ocp4.example.com ...
```

- 6. Générez une charge de mémoire supplémentaire pouvant être gérée par le conteneur.

- 6.1. Ouvrez deux fenêtres de terminal supplémentaires pour surveiller en permanence la charge de votre application. Accédez à l'API de l'application dans la première fenêtre de terminal pour simuler une sollicitation de mémoire supplémentaire sur le conteneur.
  - Dans la deuxième fenêtre de terminal, exécutez `watch oc get pods` pour surveiller continuellement l'état de chaque pod.
  - Dans la troisième fenêtre de terminal, exécutez `watch oc adm top pod` pour surveiller en continu l'utilisation de processeur et de mémoire de chaque pod.
- 6.2. Dans la première fenêtre de terminal, utilisez l'API de l'application pour augmenter la charge de mémoire de 150 Mo pendant 60 secondes. Utilisez le nom de domaine complet identifié précédemment dans la route. Pendant que vous attendez que la commande `curl` soit terminée, observez la sortie dans les deux autres fenêtres de terminal.

```
[student@workstation ~]$ curl -X GET \
>   http://loadtest.apps.ocp4.example.com/api/loadtest/v1/mem/150/60
curl: (52) Empty reply from server
```

- 6.3. Dans la deuxième fenêtre de terminal, observez la sortie de `watch oc get pods`. Étant donné que le conteneur peut gérer la charge supplémentaire, vous devez voir que l'état du pod d'application unique est `Running` pour l'ensemble de la demande `curl`.

```
Every 2.0s: oc get pods ...
...
NAME          READY   STATUS    RESTARTS   AGE
loadtest-f7495948-tlxgm  1/1     Running   0          7m34s
```

- 6.4. Dans la troisième fenêtre de terminal, observez la sortie de `watch oc adm top pod`. L'utilisation initiale de la mémoire pour le pod est d'environ 20-30 Mi.

```
Every 2.0s: oc adm top pod ...
...
NAME          CPU(cores)   MEMORY(bytes)
loadtest-f7495948-tlxgm  0m           20Mi
```

À mesure que la demande d'API est effectuée, vous devez voir l'utilisation de mémoire pour le pod croître jusqu'à environ 170-180 Mi.

```
Every 2.0s: oc adm top pod ...
```

NAME	CPU(cores)	MEMORY(bytes)
loadtest-f7495948-tlxgm	0m	172Mi

Un court instant après la fin de la demande curl, vous devez voir l'utilisation de mémoire redescendre à environ 20-30 Mi.

```
Every 2.0s: oc adm top pod ...
```

NAME	CPU(cores)	MEMORY(bytes)
loadtest-f7495948-tlxgm	0m	20Mi

- 7. Générez une charge de mémoire supplémentaire ne pouvant pas être gérée par le conteneur.

- 7.1. Utilisez l'API de l'application pour augmenter la charge de mémoire de 200 Mo pendant 60 secondes. Observez la sortie dans les deux autres fenêtres de terminal.

```
[student@workstation ~]$ curl -X GET \
>   http://loadtest.apps.ocp4.example.com/api/loadtest/v1/mem/200/60
<html><body><h1>502 Bad Gateway</h1>
The server returned an invalid or incomplete response.
</body></html>
```

- 7.2. Dans la deuxième fenêtre de terminal, observez la sortie de watch oc get pods. Presque immédiatement après avoir exécuté la commande curl, l'état du pod passera à OOMKilled. Vous pouvez même observer l'état Error. La mémoire du pod est insuffisante et doit être supprimée et redémarrée. L'état peut passer à CrashLoopBackOff avant de revenir à l'état Running. Le nombre de redémarrages s'incrémentera également.

```
Every 2.0s: oc get pods ...
```

NAME	READY	STATUS	RESTARTS	AGE
loadtest-f7495948-tlxgm	0/1	OOMKilled	0	9m13s

Dans certains cas, le pod peut avoir redémarré et être passé par l'état Running avant que vous n'ayez le temps de basculer vers la deuxième fenêtre de terminal. Le nombre de redémarrages est incrémenté de 0 à 1.

```
Every 2.0s: oc get pods ...
```

NAME	READY	STATUS	RESTARTS	AGE
loadtest-f7495948-tlxgm	1/1	Running	1	9m33s

- 7.3. Dans la troisième fenêtre de terminal, observez la sortie de watch oc adm top pod. Une fois le pod stoppé, les métriques indiquent que le pod utilise peu de ressources, voire aucune, pendant une courte période.

```
Every 2.0s: oc adm top pod      ...

NAME                  CPU(cores)   MEMORY(bytes)
loadtest-f7495948-tlxgm  8m          0Mi
```

- 7.4. Dans la première fenêtre de terminal, supprimez toutes les ressources associées à la deuxième application.

```
[student@workstation ~]$ oc delete all -l app=loadtest
pod "loadtest-f7495948-tlxgm" deleted
service "loadtest" deleted
deployment.apps "loadtest" deleted
route.route.openshift.io "loadtest" deleted
```

Dans la deuxième et la troisième fenêtres de terminal, appuyez sur **Ctrl+C** pour terminer la commande `watch`. Vous pouvez également fermer la deuxième et la troisième fenêtre de terminal.

- 8. En tant qu'administrateur de cluster, créez des quotas pour le projet `schedule-limit`.

- 8.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 8.2. Créez un quota nommé `project-quota` qui limite le projet `schedule-limit` à 3 processeurs, 1 Go de mémoire et 2 mappages de configuration.

```
[student@workstation ~]$ oc create quota project-quota \
>   --hard cpu="3",memory="1G",configmaps="2" \
>   -n schedule-limit
resourcequota/project-quota created
```



### Note

Cet exercice place un quota sur les mappages de configuration pour démontrer ce qui se passe lorsqu'un utilisateur tente de dépasser le quota.

- 9. En tant que developer, essayez de dépasser le quota de mappage de configuration pour le projet.

- 9.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.

...output omitted...
```

- 9.2. Essayez de créer un mappage de configuration. Tous les espaces de noms du cluster ont deux mappages de configuration contenant les certificats nécessaires à son fonctionnement. La création d'un mappage de configuration échoue, car un troisième mappage de configuration dépasse le quota de deux mappages.

```
[student@workstation ~]$ oc create configmap my-config
error: failed to create configmap: configmaps "my-config" is forbidden: exceeded
quota: project-quota, requested: configmaps=1, used: configmaps=2, limited:
configmaps=2
```

- 10. En tant qu'administrateur du cluster, configurez tous les nouveaux projets avec des ressources de quotas et de plages de limites par défaut.

- 10.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.

...output omitted...
```

- 10.2. Redirigez la sortie de la commande `oc adm create-bootstrap-project-template` afin de pouvoir personnaliser la création du projet.

```
[student@workstation ~]$ oc adm create-bootstrap-project-template \
>   -o yaml > /tmp/project-template.yaml
```

- 10.3. Modifiez le fichier `/tmp/project-template.yaml` pour ajouter les ressources de quotas et de plages de limites définies dans le fichier `~/D0280/labs/schedule-limit/quota-limits.yaml`. Ajoutez les lignes avant la section `parameters`.

```
...output omitted...
- apiVersion: v1
  kind: ResourceQuota
  metadata:
    name: ${PROJECT_NAME}-quota
  spec:
    hard:
      cpu: 3
      memory: 10G
- apiVersion: v1
  kind: LimitRange
  metadata:
    name: ${PROJECT_NAME}-limits
  spec:
    limits:
      - type: Container
        defaultRequest:
          cpu: 30m
          memory: 30M
  parameters:
    - name: PROJECT_NAME
    - name: PROJECT_DISPLAYNAME
```

```
- name: PROJECT_DESCRIPTION  
- name: PROJECT_ADMIN_USER  
- name: PROJECT_REQUESTING_USER
```

**Note**

Le fichier `~/DO280/solutions/schedule-limit/project-template.yaml` contient la configuration correcte et peut être utilisé à des fins de comparaison.

- 10.4. Utilisez le fichier `/tmp/project-template.yaml` pour créer une ressource de modèle dans l'espace de noms `openshift-config`.

```
[student@workstation ~]$ oc create -f /tmp/project-template.yaml \  
> -n openshift-config  
template.template.openshift.io/project-request created
```

- 10.5. Mettez à jour le cluster de manière à utiliser le modèle de projet personnalisé.

```
[student@workstation ~]$ oc edit projects.config.openshift.io/cluster
```

Modifiez la section `spec` de manière à utiliser les lignes suivantes en gras.

```
...output omitted...  
spec:  
  projectRequestTemplate:  
    name: project-request
```

Assurez-vous que la mise en retrait est correcte, puis enregistrez vos modifications.

```
project.config.openshift.io/cluster edited
```

- 10.6. Après une modification réussie, les pods `apiserver` de l'espace de noms `openshift-apiserver` sont recréés.

```
[student@workstation ~]$ watch oc get pods -n openshift-apiserver
```

Attendez que les trois nouveaux pods `apiserver` soient prêts à fonctionner.

```
Every 2.0s: oc get pods -n openshift-apiserver ...  
  
NAME           READY   STATUS    RESTARTS   AGE  
apiserver-868dccf5fb-885dz  2/2     Running   0          63s  
apiserver-868dccf5fb-8j4vh  2/2     Running   0          39s  
apiserver-868dccf5fb-r4j9b  2/2     Running   0          24s
```

Appuyez sur `Ctrl+C` pour terminer la commande `watch`.

- 10.7. Créez un projet de test pour vérifier que le modèle de projet personnalisé fonctionne comme prévu.

```
[student@workstation ~]$ oc new-project template-test
Now using project "template-test" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 10.8. Listez le quota de ressources et les ressources de plages de limites dans le projet de test.

```
[student@workstation ~]$ oc get resourcequotas,limitranges
NAME                                     AGE   REQUEST           LIMIT
resourcequota/template-test-quota     87s   cpu: 0/3, memory: 0/10G

NAME          CREATED AT
limitrange/template-test-limits    2021-06-02T15:46:37Z
```

- 11. Procédez au nettoyage de l'environnement d'atelier en supprimant le projet créé au cours de cet exercice.

- 11.1. Supprimez le projet `schedule-limit`.

```
[student@workstation ~]$ oc delete project schedule-limit
project.project.openshift.io "schedule-limit" deleted
```

- 11.2. Supprimez le projet `template-test`.

```
[student@workstation ~]$ oc delete project template-test
project.project.openshift.io "template-test" deleted
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab schedule-limit finish
```

L'exercice guidé est maintenant terminé.

# Mise à l'échelle d'une application

---

## Résultats

À la fin de cette section, vous devez pouvoir contrôler le nombre de répliques d'un pod, spécifier le nombre de répliques dans un déploiement, mettre à l'échelle manuellement le nombre de répliques et créer une ressource de mise à l'échelle automatique de pod horizontale (HPA).

## Spécification des répliques de pod dans les charges de travail de configuration

Le nombre de répliques de pod pour un déploiement spécifique peut être augmenté ou diminué pour répondre à vos besoins. Malgré les ressources `ReplicaSet` et `ReplicationController`, le nombre de répliques nécessaires pour une application est généralement défini dans une ressource de déploiement. Un ensemble de réplica ou un contrôleur de réPLICATION (géré par un déploiement) s'assure que le nombre spécifié de répliques d'un pod est exécuté à tout moment. L'ensemble de répliques ou le contrôleur de réPLICATION peuvent ajouter ou supprimer des pods selon les besoins pour se conformer au nombre de répliques souhaité.

Les ressources de déploiement contiennent :

- Le nombre de répliques souhaité
- Un sélecteur pour l'identification des pods gérés
- Une définition, ou modèle, de pod pour créer un pod répliqué (y compris les étiquettes à appliquer au pod)

La ressource de déploiement suivante (créeée à l'aide de la commande `oc create deployment`) affiche les éléments suivants :

```
apiVersion: apps/v1
kind: Deployment
...output omitted...
spec:
  replicas: 1 ①
  selector:
    matchLabels:
      deployment: scale ②
  strategy: {}
  template: ③
    metadata:
      labels:
        deployment: scale ④
  spec:
    containers:
    ...output omitted...
```

① Spécifie le nombre de copies (ou de répliques) désiré d'un pod à exécuter.

- ❷ Un ensemble de réplicas utilise un sélecteur pour compter le nombre de pods en cours d'exécution, de la même façon qu'un service utilise un sélecteur pour trouver les pods à des fins d'équilibrage de charge.
- ❸ Un modèle pour les pods que l'ensemble de répliques ou le contrôleur de réPLICATION crée.
- ❹ Les libellés appliqués aux pods créés par le modèle doivent correspondre à ceux utilisés pour le sélecteur.

Si la ressource de déploiement est sous contrôle de version, modifiez la ligne `replicas` dans le fichier de ressources et appliquez les modifications à l'aide de la commande `oc apply`.

Dans une ressource de déploiement, un sélecteur est un ensemble de libellés auquel tous les pods gérés par l'ensemble de réplicas doivent correspondre. Le même ensemble de libellés doit être inclus dans la définition de pod instanciée par le déploiement. Ce sélecteur est utilisé afin de déterminer combien d'instances du pod sont déjà en cours d'exécution et ajuster le nombre selon les besoins.



#### Note

Le contrôleur de réPLICATION n'effectue pas de mise à l'échelle automatique, car il ne surveille ni la charge ni le trafic. La ressource de mise à l'échelle automatique de pod horizontale, qui sera présentée ultérieurement dans cette section, gère la mise à l'échelle automatique.

## Mise à l'échelle manuelle du nombre de répliques de pod

Les développeurs et administrateurs peuvent choisir de mettre à l'échelle manuellement le nombre de répliques de pods d'un projet. Des pods supplémentaires peuvent être nécessaires pour une augmentation anticipée du trafic, ou le nombre de pods peut être réduit pour récupérer des ressources qui peuvent être utilisées ailleurs par le cluster. Qu'il s'agisse d'augmenter ou de réduire le nombre de répliques de pods, la première étape consiste à identifier le déploiement approprié à mettre à l'échelle à l'aide de la commande `oc get deployment` :

```
[user@host ~]$ oc get deployment
NAME      READY     UP-TO-DATE   AVAILABLE   AGE
scale     1/1       1           1           8h
```

Le nombre de répliques d'une ressource de déploiement peut être modifié manuellement à l'aide de la commande `oc scale` :

```
[user@host ~]$ oc scale --replicas 5 deployment/scale
```

La ressource de déploiement propage la modification à l'ensemble de répliques ; il réagit à la modification en créant de nouveaux pods (répliques) ou en supprimant ceux existants, selon que le nouveau nombre de répliques souhaité est inférieur ou supérieur au nombre existant.

Bien qu'il soit possible de manipuler directement la ressource de l'ensemble de répliques, la pratique recommandée est de manipuler la ressource de déploiement. Un nouveau déploiement crée un nouvel ensemble de répliques ou un nouveau contrôleur de réPLICATION, et les modifications apportées directement à un ensemble de répliques ou à un contrôleur de réPLICATION antérieur sont ignorées.

## Mise à l'échelle automatique de pods

OpenShift peut mettre à l'échelle automatiquement un déploiement en fonction de la charge actuelle des pods de l'application au moyen d'une ressource de type `HorizontalPodAutoscaler`.

Une ressource de mise à l'échelle automatique de pod horizontale utilise les mesures de performance collectées par le sous-système OpenShift Metrics. Le sous-système Metrics est pré-installé dans OpenShift 4, et ne nécessite pas d'installation séparée, comme dans OpenShift 3. Pour mettre à l'échelle automatiquement un déploiement, vous devez spécifier des demandes de ressources pour les pods afin que la mise à l'échelle automatique de pod horizontale puisse calculer le pourcentage d'utilisation.

Pour créer une ressource de mise à l'échelle automatique de pod horizontale, il est recommandé d'utiliser la commande `oc autoscale`, par exemple :

```
[user@host ~]$ oc autoscale deployment/hello \
>   --min 1 --max 10 --cpu-percent 80
```

La commande précédente crée une ressource de mise à l'échelle automatique de pod horizontale qui modifie le nombre de répliques sur le déploiement `hello` pour maintenir ses pods en dessous de 80 % de leur utilisation de processeur totale requise.

La commande `oc autoscale` crée une ressource de mise à l'échelle automatique de pod horizontale utilisant le déploiement comme argument (`hello` dans l'exemple précédent).

Les valeurs maximale et minimale pour la ressource de mise à l'échelle automatique de pod horizontale servent à prendre en charge les pics de charge et à éviter de surcharger le cluster OpenShift. Si la charge de l'application change trop rapidement, il peut alors s'avérer judicieux de conserver un certain nombre de pods pour faire face aux pics soudains de demandes d'utilisateurs. Inversement, un nombre trop élevé de pods peut utiliser toute la capacité du cluster et affecter les autres applications partageant le même cluster OpenShift.

Pour obtenir des informations sur les ressources de mise à l'échelle automatique de pod horizontale dans le projet en cours, utilisez les commandes `oc get`. Par exemple :

```
[user@host ~]$ oc get hpa
NAME      REFERENCE          TARGETS      MINPODS   MAXPODS   REPLICAS   ...
hello    Deployment/hello    <unknown>/80%    1          10        1          ...
scale    Deployment/scale    60%/80%       2          10        2          ...
```



### Important

La mise à l'échelle automatique de pod horizontale a initialement la valeur <unknown> dans la colonne TARGET. Il faudra peut-être attendre jusqu'à cinq minutes pour que <unknown> soit remplacé par un pourcentage de l'utilisation actuelle.

La valeur persistante de <unknown> dans la colonne TARGETS peut indiquer que le déploiement ne définit pas les demandes de ressources pour la métrique. La mise à l'échelle automatique de pod horizontale ne mettra pas ces pods à l'échelle.

La plupart des pods créés à l'aide de la commande `oc new-app` ne définissent pas les demandes de ressources. Ici, l'utilisation de la mise à l'échelle automatique d'OpenShift peut, par conséquent, nécessiter de modifier les ressources de déploiement, de créer des fichiers de ressources YAML ou JSON personnalisés pour votre application ou d'ajouter des ressources de plage de limite à votre projet qui définissent les demandes de ressource par défaut.



### Références

Pour plus d'informations, reportez-vous à la section *Automatically scaling pods with the Horizontal Pod Autoscaler* du chapitre *Working with pods* de la documentation *Nodes* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/nodes/index#nodes-pods-autoscaling](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/nodes/index#nodes-pods-autoscaling)

## ► Exercice guidé

# Mise à l'échelle d'une application

Au cours de cet exercice, vous allez mettre à l'échelle une application manuellement et automatiquement.

## Résultats

Vous serez en mesure d'utiliser l'interface de ligne de commande OpenShift pour :

- Mettre à l'échelle manuellement une application.
- Configurer une application pour qu'elle soit automatiquement mise à l'échelle en fonction de l'utilisation.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande permet de s'assurer que l'API du cluster est accessible et crée les fichiers de ressources que vous utiliserez au cours de l'activité.

```
[student@workstation ~]$ lab schedule-scale start
```

## Instructions

- 1. En tant qu'utilisateur `developer`, créez un nouveau projet nommé `schedule-scale`.
- 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `developer`.

```
[student@workstation ~]$ oc login -u developer -p developer \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Créez un nouveau projet pour cet exercice guidé nommé `schedule-scale`.

```
[student@workstation ~]$ oc new-project schedule-scale
Now using project "schedule-scale" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 2. Déployez une application de test pour cet exercice qui demande explicitement des ressources de conteneur pour le processeur et la mémoire.
- 2.1. Modifiez le fichier de ressources situé sur `~/D0280/labs/schedule-scale/loadtest.yaml` pour définir à la fois les demandes et les limites de processeur et d'utilisation de la mémoire.

```
[student@workstation ~]$ vim ~/DO280/labs/schedule-scale/loadtest.yaml
```

- 2.2. Remplacez la ligne `resources: {}` par les lignes mises en surbrillance listées ci-dessous. Assurez-vous que la mise en retrait est bonne avant d'enregistrer le fichier.

```
...output omitted...
spec:
  containers:
    - image: quay.io/redhattraining/loadtest:v1.0
      name: loadtest
      resources:
        requests:
          cpu: "25m"
          memory: 25Mi
        limits:
          cpu: "100m"
          memory: 100Mi
  status: {}
```

- 2.3. Créez la nouvelle application à l'aide de votre fichier de ressources.

```
[student@workstation ~]$ oc create --save-config \
>   -f ~/DO280/labs/schedule-scale/loadtest.yaml
deployment.apps/loadtest created
service/loadtest created
route.route.openshift.io/loadtest created
```

- 2.4. Vérifiez que l'état du pod d'application est `Running`. Il se peut que vous deviez exécuter plusieurs fois la commande `oc get pods`.

```
[student@workstation ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
loadtest-5f9565dbfb-jm9md   1/1     Running   0          23s
```

- 2.5. Vérifiez que votre pod d'application spécifie les demandes et limites des ressources.

```
[student@workstation ~]$ oc describe pod/loadtest-5f9565dbfb-jm9md | \
>   grep -A2 -E "Limits|Requests"
Limits:
  cpu:      100m
  memory:   100Mi
Requests:
  cpu:      25m
  memory:   25Mi
```

- 3. Mettez à l'échelle manuellement le déploiement `loadtest` en augmentant d'abord, puis en réduisant le nombre de pods en cours d'exécution.

- 3.1. Mettez à l'échelle le déploiement `loadtest` jusqu'à cinq pods.

```
[student@workstation ~]$ oc scale --replicas 5 deployment/loadtest
deployment.apps/loadtest scaled
```

- 3.2. Vérifiez que les cinq pods d'application sont en cours d'exécution. Il se peut que vous deviez exécuter plusieurs fois la commande `oc get pods`.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
loadtest-5f9565dbfb-22f9s  1/1     Running   0          54s
loadtest-5f9565dbfb-8l2rn  1/1     Running   0          54s
loadtest-5f9565dbfb-jm9md  1/1     Running   0          3m17s
loadtest-5f9565dbfb-lfhns  1/1     Running   0          54s
loadtest-5f9565dbfb-prjkl  1/1     Running   0          54s
```

- 3.3. Mettez à l'échelle le déploiement `loadtest` à un pod.

```
[student@workstation ~]$ oc scale --replicas 1 deployment/loadtest
deployment.apps/loadtest scaled
```

- 3.4. Vérifiez qu'un seul pod d'application est en cours d'exécution. Vous devrez peut-être exécuter plusieurs fois la commande `oc get pods` en attendant que les autres pods prennent fin.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
loadtest-5f9565dbfb-prjkl  1/1     Running   0          72s
```

- ▶ 4. Configurez l'application `loadtest` pour qu'elle soit automatiquement mise à l'échelle en fonction de la charge, puis testez l'application en appliquant la charge.

- 4.1. Créez une mise à l'échelle automatique de pod horizontale qui garantit que l'application `loadtest` a toujours 2 pods d'application en cours d'exécution ; ce nombre peut être augmenté jusqu'à un maximum de 10 pods lorsque la charge processeur dépasse 50 %.

```
[student@workstation ~]$ oc autoscale deployment/loadtest \
>   --min 2 --max 10 --cpu-percent 50
horizontalpodautoscaler.autoscaling/loadtest autoscaled
```

- 4.2. Attendez que la mise à l'échelle automatique de pod `loadtest` horizontale signale l'utilisation dans la colonne TARGETS.

```
[student@workstation ~]$ watch oc get hpa/loadtest
```

Appuyez sur `Ctrl+C` pour quitter la commande `watch` une fois que la valeur `<unknown>` s'est transformée en pourcentage.

```
Every 2.0s: oc get hpa/loadtest ...
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	...
loadtest	Deployment/loadtest	0%/50%	2	10	2	...

**Note**

Cela peut prendre jusqu'à cinq minutes pour que l'entrée <unknown> dans la colonne TARGETS passe à 0%. Si l'entrée <unknown> ne change pas, utilisez la commande `oc describe` pour vérifier que les conteneurs de l'application `loadtest` demandent des ressources processeur.

- 4.3. L'image de conteneur `loadtest` est conçue pour augmenter la charge processeur ou mémoire sur le conteneur en effectuant une demande à l'API de l'application. Identifiez le nom de domaine complet utilisé dans la route.

```
[student@workstation ~]$ oc get route/loadtest
NAME      HOST/PORT
loadtest  loadtest-schedule-scale.apps.ocp4.example.com ...
```

- 4.4. Accédez à l'API de l'application pour simuler une sollicitation processeur supplémentaire sur le conteneur. Passez à l'étape suivante tout en attendant que la commande `curl` soit terminée.

```
[student@workstation ~]$ curl -X GET \
>   http://loadtest-schedule-scale.apps.ocp4.example.com/api/loadtest/v1/cpu/1
curl: (52) Empty reply from server
```

- 4.5. Ouvrez une deuxième fenêtre de terminal et surveillez continuellement l'état de la mise à l'échelle automatique de pod horizontale.

**Note**

L'augmentation de l'activité de l'application ne déclenche pas immédiatement la mise à l'échelle automatique. Patientez quelques instants si vous ne voyez aucune modification du nombre de répliques.

```
[student@workstation ~]$ watch oc get hpa/loadtest
```

Au fur et à mesure que la charge augmente (visible dans la colonne TARGETS), vous devriez voir le nombre sous REPLICAS augmenter. Observez la sortie pendant une ou deux minutes avant de passer à l'étape suivante. Votre sortie sera probablement différente de ce qui est affiché ci-dessous.

```
Every 2.0s: oc get hpa/loadtest ...
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	...
loadtest	Deployment/loadtest	172%/50%	2	10	9	...

**Note**

Bien que la démultiplication de la ressource `autoscaler` de pod horizontale puisse être rapide, la réduction est plus lente. Au lieu d'attendre que l'application `loadtest` ne revienne à deux pods, poursuivez avec le reste de l'exercice.

- ▶ 5. Dans la première fenêtre de terminal, créez une seconde application nommée `scaling`. Mettez à l'échelle l'application, puis vérifiez les réponses provenant des pods d'application.
  - 5.1. Créez une nouvelle application avec la commande `oc new-app` à l'aide de l'image de conteneur située dans `quay.io/redhattraining/scaling:v1.0`.

```
[student@workstation ~]$ oc new-app --name scaling \
>   --image quay.io/redhattraining/scaling:v1.0
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "scaling" created
  deployment.apps "scaling" created
  service "scaling" created
--> Success
  Application is not exposed.
  You can expose services to the outside world by executing one or more of the
  commands below:
    'oc expose svc/scaling'
  Run 'oc status' to view your app.
```

- 5.2. Créez une route vers l'application en exposant le service à l'application.

```
[student@workstation ~]$ oc expose svc/scaling
route.route.openshift.io/scaling exposed
```

- 5.3. Mettez à l'échelle l'application jusqu'à trois pods à l'aide de la ressource de déploiement de l'application.

```
[student@workstation ~]$ oc scale --replicas 3 deployment/scaling
deployment.apps/scaling scaled
```

- 5.4. Vérifiez que les trois pods de l'application `scaling` sont en cours d'exécution et identifiez leurs adresses IP associées.

```
[student@workstation ~]$ oc get pods -o wide -l deployment=scaling
NAME          READY   STATUS    RESTARTS   AGE     IP           NODE   ...
scaling-1-bm4m2 1/1     Running   0          45s    10.10.0.29  master01 ...
scaling-1-w7whl 1/1     Running   0          45s    10.8.0.45   master03 ...
scaling-1-xqvs2 1/1     Running   0          6m1s   10.9.0.58   master02 ...
```

- 5.5. Affichez le nom d'hôte utilisé pour acheminer les demandes vers l'application `scaling`.

```
[student@workstation ~]$ oc get route/scaling
NAME      HOST/PORT
scaling   scaling-schedule-scale.apps.ocp4.example.com ...
```

- 5.6. Lorsque vous accédez au nom d'hôte de votre application, la page PHP affiche l'adresse IP du pod qui a répondu à la requête. Envoyez plusieurs demandes à votre application, puis triez les réponses pour compter le nombre de demandes envoyées à chaque pod. Exécutez le script situé à l'adresse ~/DO280/labs/schedule-scale/curl-route.sh.

```
[student@workstation ~]$ ~/DO280/labs/schedule-scale/curl-route.sh
34 Server IP: 10.10.0.29
34 Server IP: 10.8.0.45
32 Server IP: 10.9.0.58
```

- 6. **Optional:** Check the status of the horizontal pod autoscaler running for the `loadtest` application.

Si la commande `watch oc get hpa/loadtest` est toujours exécutée dans la deuxième fenêtre de terminal, basculez vers celle-ci pour observer la sortie. Si suffisamment de temps s'est écoulé, le nombre de répliques doit à nouveau être de deux.

```
Every 2.0s: oc get hpa/loadtest ...
NAME      REFERENCE          TARGETS  MINPODS  MAXPODS  REPLICAS  ...
loadtest  Deployment/loadtest  0%/50%    2         10        2         ...
```

Lorsque vous avez terminé, appuyez sur `Ctrl+C` pour quitter la commande `watch`, puis fermez la deuxième fenêtre de terminal.

- 7. Procédez au nettoyage de l'environnement d'atelier en supprimant le projet `schedule-scale`:

```
[student@workstation ~]$ oc delete project schedule-scale
project.project.openshift.io "schedule-scale" deleted
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab schedule-scale finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Contrôle de la planification des pods

### Liste de contrôle des performances

Dans cet atelier, vous allez configurer une application pour qu'elle s'exécute sur un sous-ensemble de nœuds de cluster et pour qu'elle se mette à l'échelle avec la charge.

### Résultats

Vous serez en mesure d'utiliser l'interface de ligne de commande OpenShift pour :

- Ajouter une nouvelle étiquette aux nœuds.
- Déployer des pods sur les nœuds correspondant à une étiquette spécifique.
- Demander des ressources processeur et mémoire pour les pods.
- Configurer une application pour qu'elle soit automatiquement mise à l'échelle.
- Créer un quota pour limiter la quantité de ressources pouvant être consommées par un projet.

### Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande assure que l'API du cluster est accessible et crée un répertoire pour les fichiers de l'exercice.

```
[student@workstation ~]$ lab schedule-review start
```

### Instructions

1. En tant qu'utilisateur `admin`, étiquetez les deux nœuds avec l'étiquette `tier`. Attribuez au nœud `master01` l'étiquette `tier=gold` et au nœud `master02` l'étiquette `tier=silver`.
2. Basculez en utilisateur `developer` et créez un nouveau projet nommé `schedule-review`.
3. Créez une nouvelle application nommée `loadtest` à l'aide de l'image de conteneur située dans `quay.io/redhattraining/loadtest:v1.0`.
  - L'application `loadtest` doit être déployée sur les nœuds étiquetés avec `tier=silver`.
  - Assurez-vous que chaque conteneur demande `100m` de processeur et `20Mi` de mémoire.
4. Créez une route vers votre application nommée `loadtest` à l'aide du nom d'hôte par défaut (généré automatiquement). En fonction de la manière dont vous avez créé votre application, vous devrez peut-être créer un service avant de créer la route. Votre application fonctionne comme prévu si l'URL suivante renvoie `{"health": "ok"}`.
  - `http://loadtest-schedule-review.apps.ocp4.example.com/api/loadtest/v1/healthz`

5. Créez une mise à l'échelle automatique de pod horizontale appelée `loadtest` pour l'application `loadtest`, qui mettra à l'échelle de 2 pods à un maximum de 40 pods si la charge processeur dépasse 70%. Vous pouvez tester la mise à l'échelle automatique de pod horizontale en accédant à l'URL suivante pour déclencher les actions de test de charge.
- `http://loadtest-schedule-review.apps.ocp4.example.com/api/loadtest/v1/cpu/3`



#### Note

Bien que la mise à l'échelle automatique de pod horizontale permette de mettre à l'échelle l'application `loadtest`, votre cluster OpenShift arrivera à court de ressources avant d'atteindre un maximum de 40 pods.

L'URL du déclencheur de test de charge ne renvoie aucune sortie.

6. En tant qu'utilisateur `admin`, implémentez un quota nommé `review-quota` sur le projet `schedule-review`. Limitez le projet `schedule-review` à un maximum de 1 processeur complet, 2G de mémoire et 20 pods.

## Évaluation

Exécutez la commande `lab` suivante pour contrôler votre travail. Si la commande `lab` signale des erreurs, passez en revue vos modifications, apportez des corrections, et exécutez à nouveau la commande `lab` tant que des erreurs persistent.

```
[student@workstation ~]$ lab schedule-review grade
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab schedule-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Contrôle de la planification des pods

### Liste de contrôle des performances

Dans cet atelier, vous allez configurer une application pour qu'elle s'exécute sur un sous-ensemble de nœuds de cluster et pour qu'elle se mette à l'échelle avec la charge.

### Résultats

Vous serez en mesure d'utiliser l'interface de ligne de commande OpenShift pour :

- Ajouter une nouvelle étiquette aux nœuds.
- Déployer des pods sur les nœuds correspondant à une étiquette spécifique.
- Demander des ressources processeur et mémoire pour les pods.
- Configurer une application pour qu'elle soit automatiquement mise à l'échelle.
- Créer un quota pour limiter la quantité de ressources pouvant être consommées par un projet.

### Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande assure que l'API du cluster est accessible et crée un répertoire pour les fichiers de l'exercice.

```
[student@workstation ~]$ lab schedule-review start
```

### Instructions

1. En tant qu'utilisateur `admin`, étiquetez les deux nœuds avec l'étiquette `tier`. Attribuez au nœud `master01` l'étiquette `tier=gold` et au nœud `master02` l'étiquette `tier=silver`.
  - 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
>   https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Identifiez si des nœuds utilisent déjà l'étiquette `tier`.

```
[student@workstation ~]$ oc get nodes -L tier
NAME      STATUS    ROLES          AGE     VERSION        TIER
master01   Ready     master,worker  12d    v1.23.3+e419edf
master02   Ready     master,worker  12d    v1.23.3+e419edf
master03   Ready     master,worker  12d    v1.23.3+e419edf
```

- 1.3. Étiquetez le nœud master01 avec l'étiquette tier=gold.

```
[student@workstation ~]$ oc label node master01 tier=gold
node/master01 labeled
```

- 1.4. Étiquetez le nœud master02 avec l'étiquette tier=silver.

```
[student@workstation ~]$ oc label node master02 tier=silver
node/master02 labeled
```

- 1.5. Contrôlez que les étiquettes ont été correctement ajoutées.

```
[student@workstation ~]$ oc get nodes -L tier
NAME      STATUS    ROLES          AGE     VERSION        TIER
master01   Ready     master,worker  12d    v1.23.3+e419edf  gold
master02   Ready     master,worker  12d    v1.23.3+e419edf  silver
master03   Ready     master,worker  12d    v1.23.3+e419edf
```

2. Basculez en utilisateur developer et créez un nouveau projet nommé schedule-review.

- 2.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur developer.

```
[student@workstation ~]$ oc login -u developer -p developer
Login successful.
...output omitted...
```

- 2.2. Créez le projet schedule-review.

```
[student@workstation ~]$ oc new-project schedule-review
Now using project "schedule-review" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

3. Créez une nouvelle application nommée loadtest à l'aide de l'image de conteneur située dans quay.io/redhattraining/loadtest:v1.0.

- L'application loadtest doit être déployée sur les nœuds étiquetés avec tier=silver.
- Assurez-vous que chaque conteneur demande 100m de processeur et 20Mi de mémoire.

- 3.1. Afin de faciliter les réglages à venir, créez un fichier de ressources de déploiement sans réellement créer l'application.

```
[student@workstation ~]$ oc create deployment loadtest --dry-run=client \
>   --image quay.io/redhattraining/loadtest:v1.0 \
>   -o yaml > ~/D0280/labs/schedule-review/loadtest.yaml
```

**chapitre 13 |** Contrôle de la planification des pods

- 3.2. Modifiez ~/D0280/labs/schedule-review/loadtest.yaml pour spécifier un sélecteur de nœud.

```
[student@workstation ~]$ vim ~/D0280/labs/schedule-review/loadtest.yaml
```

Ajoutez les lignes mises en surbrillance listées ci-dessous et assurez-vous que la mise en retrait est bonne.

```
...output omitted...
spec:
  nodeSelector:
    tier: silver
  containers:
    - image: quay.io/redhattraining/loadtest:v1.0
      name: loadtest
      resources: {}
status: {}
```

- 3.3. Continuez à modifier ~/D0280/labs/schedule-review/loadtest.yaml.

Remplacez la ligne **resources: {}** par les lignes mises en surbrillance listées ci-dessous. Assurez-vous que la mise en retrait est bonne avant d'enregistrer le fichier.

```
...output omitted...
spec:
  nodeSelector:
    tier: silver
  containers:
    - image: quay.io/redhattraining/loadtest:v1.0
      name: loadtest
      resources:
        requests:
          cpu: "100m"
          memory: 20Mi
status: {}
```

- 3.4. Créez l'application loadtest.

```
[student@workstation ~]$ oc create --save-config \
>   -f ~/D0280/labs/schedule-review/loadtest.yaml
deployment.apps/loadtest created
```

- 3.5. Vérifiez qu'un seul pod d'application est en cours d'exécution. Il se peut que vous deviez exécuter plusieurs fois la commande `oc get pods`.

```
[student@workstation ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
loadtest-85f7669897-z4mq7   1/1     Running   0          53s
```

- 3.6. Vérifiez que votre pod d'application spécifie les demandes de ressources.

```
[student@workstation ~]$ oc describe pod/loadtest-85f7669897-z4mq7 | \
>     grep -A2 Requests
Requests:
cpu:        100m
memory:    20Mi
```

4. Créez une route vers votre application nommée `loadtest` à l'aide du nom d'hôte par défaut (généré automatiquement). En fonction de la manière dont vous avez créé votre application, vous devrez peut-être créer un service avant de créer la route. Votre application fonctionne comme prévu si l'URL suivante renvoie {"`health`": "ok"}.

- `http://loadtest-schedule-review.apps.ocp4.example.com/api/loadtest/v1/healthz`

- 4.1. Créez un service en exposant le déploiement pour l'application `loadtest`.

```
[student@workstation ~]$ oc expose deployment/loadtest \
>     --port 80 --target-port 8080
service/loadtest exposed
```

- 4.2. Créez une route nommée `loadtest` en exposant le service `loadtest`.

```
[student@workstation ~]$ oc expose service/loadtest --name loadtest
route.route.openshift.io/loadtest exposed
```

- 4.3. Identifiez le nom d'hôte créé par la route `loadtest`.

```
[student@workstation ~]$ oc get route/loadtest
NAME      HOST/PORT          ...
loadtest   loadtest-schedule-review.apps.ocp4.example.com ...
```

- 4.4. Vérifiez l'accès à l'application `loadtest` à l'aide du nom d'hôte identifié à l'étape précédente.

```
[student@workstation ~]$ curl \
> http://loadtest-schedule-review.apps.ocp4.example.com/api/loadtest/v1/healthz
{"health": "ok"}
```

5. Créez une mise à l'échelle automatique de pod horizontale appelée `loadtest` pour l'application `loadtest`, qui mettra à l'échelle de 2 pods à un maximum de 40 pods si la charge processeur dépasse 70%. Vous pouvez tester la mise à l'échelle automatique de pod horizontale en accédant à l'URL suivante pour déclencher les actions de test de charge.

- `http://loadtest-schedule-review.apps.ocp4.example.com/api/loadtest/v1/cpu/3`

**Note**

Bien que la mise à l'échelle automatique de pod horizontale permette de mettre à l'échelle l'application `loadtest`, votre cluster OpenShift arrivera à court de ressources avant d'atteindre un maximum de 40 pods.

L'URL du déclencheur de test de charge ne renvoie aucune sortie.

- Créez la mise à l'échelle automatique de pod horizontale pour l'application `loadtest`.

```
[student@workstation ~]$ oc autoscale deployment/loadtest --name loadtest \
>   --min 2 --max 40 --cpu-percent 70
horizontalpodautoscaler.autoscaling/loadtest autoscaled
```

- Attendez que la ressource de mise à l'échelle automatique de pod `loadtest` horizontale signale l'utilisation par défaut dans la colonne TARGETS.

```
[student@workstation ~]$ watch oc get hpa/loadtest
```

Appuyez sur `Ctrl+C` pour quitter la commande `watch` une fois que la valeur `<unknown>` est passée à 0%.

```
Every 2.0s: oc get hpa/loadtest ...
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	...
loadtest	Deployment/loadtest	0%/70%	2	40	2	...

**Note**

Cela peut prendre jusqu'à cinq minutes pour que l'entrée `<unknown>` dans la colonne TARGETS passe à 0%.

Si l'entrée `<unknown>` ne change pas, utilisez la commande `oc describe` pour vérifier que les conteneurs de l'application `loadtest` demandent des ressources processeur.

- Testez la mise à l'échelle automatique de pod horizontale en appliquant la charge processeur. Utilisez le nom d'hôte précédemment identifié pour la route `loadtest`. Attendez que la commande `curl` se termine.

```
[student@workstation ~]$ curl \
> http://loadtest-schedule-review.apps.ocp4.example.com/api/loadtest/v1/cpu/3
```

**Note**

Cette commande ne génère aucune sortie.

- Vérifiez que des pods supplémentaires ont été ajoutés. Exécutez plusieurs fois la commande `oc get hpa/loadtest` jusqu'à ce que les modifications soient appliquées. Votre sortie sera probablement différente, mais vérifiez que le nombre de répliques est supérieur à 2.

```
[student@workstation ~]$ oc get hpa/loadtest
NAME      REFERENCE      TARGETS      MINPODS      MAXPODS      REPLICAS      ...
loadtest   Deployment/loadtest  1043%/70%    2            40           21           ...
```

6. En tant qu'utilisateur `admin`, implémentez un quota nommé `review-quota` sur le projet `schedule-review`. Limitez le projet `schedule-review` à un maximum de 1 processeur complet, 2G de mémoire et 20 pods.

6.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat
Login successful.
...output omitted...
```

6.2. Créez le quota de ressources.

```
[student@workstation ~]$ oc create quota review-quota \
>   --hard cpu="1",memory="2G",pods="20"
resourcequota/review-quota created
```



### Note

Le quota n'aura pas d'impact sur les pods existants, mais le planificateur évaluera le quota si de nouvelles ressources, telles que des pods, sont demandées.

## Évaluation

Exécutez la commande `lab` suivante pour contrôler votre travail. Si la commande `lab` signale des erreurs, passez en revue vos modifications, apportez des corrections, et exécutez à nouveau la commande `lab` tant que des erreurs persistent.

```
[student@workstation ~]$ lab schedule-review grade
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab schedule-review finish
```

L'atelier est maintenant terminé.

## Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Le planificateur de pod par défaut utilise des régions et des zones pour atteindre performance et redondance.
- L'étiquetage des nœuds et l'utilisation de sélecteurs de nœud influencent le placement des pods.
- Les demandes de ressources définissent le nombre minimal de ressources dont un pod a besoin pour être planifié.
- Les quotas limitent la quantité de ressources qu'un projet est autorisé à utiliser.
- Un modèle de projet personnalisé peut créer automatiquement des quotas et des plages de limites pour les nouveaux projets.
- La commande `oc scale` met à l'échelle manuellement le nombre de répliques d'un pod.
- Les mises à l'échelle automatique de pod horizontales redimensionnent dynamiquement les répliques de pods en fonction de la charge.



# Description des mises à jour du cluster

## Objectif

Décrire comment effectuer une mise à jour du cluster.

## Résultats

Décrire le processus de mise à jour du cluster.

## Sections

Description du processus de mise à jour du cluster  
(avec quiz)

# Description du processus de mise à jour du cluster

## Résultats

À la fin de cette section, vous devez pouvoir décrire le processus de mise à jour du cluster.

## Présentation des mises à jour du cluster

Red Hat OpenShift Container Platform 4 ajoute de nombreuses nouvelles fonctionnalités grâce à Red Hat Enterprise Linux CoreOS. Red Hat a publié un nouveau système de distribution de logiciels qui fournit la meilleure procédure de mise à niveau de votre cluster et système d'exploitation sous-jacent. Ce nouveau système de distribution constitue l'un des avantages significatifs des modifications de l'architecture OpenShift 4, ce qui permet aux clusters d'être mis à niveau Over-the-Air (OTA).

Ce système de distribution de logiciels pour OTA gère les manifestes de contrôleur, les rôles du cluster et toutes les autres ressources nécessaires à la mise à jour d'un cluster vers une version donnée.

Cette fonctionnalité permet de s'assurer qu'un cluster exécute la dernière version disponible de façon fluide. L'OTA permet également à un cluster d'utiliser les nouvelles fonctionnalités dès qu'elles sont disponibles, y compris les derniers correctifs de bogues et les correctifs de sécurité. L'OTA réduit considérablement les temps d'arrêt entraînés par les mises à niveau.

Red Hat héberge et gère ce service sur <https://console.redhat.com/openshift>, et héberge des images de cluster sur <https://quay.io/>.



### Important

Depuis OpenShift 4.10, le système OTA nécessite une connexion continue à Internet. Il n'est pas possible de déployer cette fonction sur des environnements déconnectés.

Pour plus d'informations sur la façon de mettre à jour les clusters déconnectés, consultez le guide *Update* et le chapitre *Installation configuration* répertorié dans la section Références.

L'OTA permet d'accélérer les mises à jour en permettant l'omission des versions intermédiaires. Par exemple, vous pouvez effectuer une mise à jour de 4.10.1 à 4.10.3, ignorant ainsi 4.10.2.

Vous utilisez une interface unique (<https://console.redhat.com/openshift>) pour gérer le cycle de vie de tous vos clusters OpenShift.

The screenshot shows the Red Hat Hybrid Cloud Console interface. On the left, there's a sidebar with 'OpenShift' at the top, followed by 'Clusters', 'Overview', 'Releases', 'Downloads', 'Insights', 'Advisor', 'Subscriptions', 'Cost Management', 'Support Cases', 'Cluster Manager Feedback', 'Red Hat Marketplace', and 'Documentation'. The main area is titled 'Clusters' and contains a table with columns: Name, Status, Type, Created, Version, Provider, and three dots. There are five rows in the table, each with a green checkmark icon and the word 'Ready' under the 'Status' column. The 'Type' column shows 'OCP' for all. The 'Created' column lists dates like '4.8.22', '4.6.52', '4.9.18', and '4.6.52'. The 'Version' column shows 'Evaluation expired' or 'days left' (e.g., '3 days left'). The 'Provider' column shows 'OpenStack' or 'AWS (us-east-2)'. Each row has a blue 'Update' button next to the provider name.

Figure 14.1: Gestion des clusters sur cloud.redhat.com

Le service définit les *procédures de mise à niveau* qui correspondent à l'éligibilité du cluster pour certaines mises à jour.

Les scénarios de mise à niveau appartiennent aux canaux de mise à jour. Un canal peut être visualisé comme une représentation de la procédure de mise à niveau. Le canal contrôle la fréquence et la stabilité des mises à jour. Le moteur de règles OTA représente les canaux sous la forme d'une série de pointeurs vers des versions données dans la procédure de mise à niveau.

Un nom de canal se compose de trois parties : le niveau (release candidate, fast et stable), la version majeure (4) et la version mineure (. 2). Exemples de noms de canaux : **stable-4.10**, **fast-4.10**, **eus-4.10** et **candidate-4.10**. Chaque canal fournit des correctifs pour une version de cluster donnée.

## Description du canal Candidate

Le canal *candidate* fournit des mises à jour pour tester l'acceptation des fonctions dans la prochaine version d'OpenShift Container Platform. Les versions « release candidate » peuvent faire l'objet de vérifications supplémentaires et sont promues au rang de canaux « fast » ou « stable » lorsqu'elles répondent aux normes de qualité.



### Note

Les mises à jour listées dans le canal *candidate* ne sont pas prises en charge par Red Hat.

## Description du canal fast

Le canal *fast* propose les mises à jour dès qu'elles sont disponibles. Ce canal convient aux environnements de développement et d'assurance qualité (QA). Vous pouvez utiliser le

canal **fast -4.10** pour effectuer une mise à niveau à partir d'une version mineure antérieure d'OpenShift Container Platform.



### Note

Les clients peuvent aider à améliorer OpenShift en rejoignant le programme Red Hat Connected Customers. Si vous rejoignez ce programme, votre cluster est enregistré sur le canal fast.

## Description du canal stable

Le canal stable contient des mises à jour différencées, ce qui signifie qu'il ne propose que des mises à jour mineures pour une version de cluster donnée et qu'il convient mieux pour les environnements de production.

Les équipes d'assistance technique et de SRE (Site Reliability Engineering) Red Hat surveillent les clusters opérationnels dotés des nouvelles mises à jour fast. Si les clusters opérationnels passent les phases de tests et validations, les mises à jour du canal fast sont activées sur le canal stable.

Si Red Hat observe des problèmes d'exploitation sur une mise à jour du canal fast, cette mise à jour est ignorée sur le canal stable. Le délai du canal stable offre le temps nécessaire pour observer les problèmes imprévus dans les clusters OpenShift réels que les tests n'ont pas révélés.

## Description du canal de prise en charge prolongée des mises à jour

Le canal de prise en charge prolongée des mises à jour (EUS) est proposé dans certaines versions mineures d'OpenShift Container Platform aux clients disposant d'abonnements Premium. Les versions d'EUS étendent la phase de maintenance à 18 mois. Il n'existe aucune différence entre les canaux **stable -4.10** et **eus -4.10**, et vous pouvez basculer vers le canal EUS dès qu'il est disponible.

À partir de la version 4.8 d'OpenShift Container Platform, Red Hat désignera toutes les versions mineures paires (par exemple, 4.8, 4.10, 4.12) comme étant des versions EUS (Extended Update Support).



### Note

Une fois la mise à niveau effectuée vers une version exclusive au canal EUS, ce cluster n'est plus éligible aux mises à niveau mineures de version tant que les mises à niveau de la prochaine version EUS ne sont pas disponibles.

## Description des chemins de mise à niveau

Cette section décrit la manière dont ces chemins de mise à niveau s'appliquent à Red Hat OpenShift Container Platform version 4.10 :

- Lorsque vous utilisez le canal **stable -4.10**, vous pouvez mettre à niveau votre cluster de la version 4.10.0 vers 4.10.1 ou 4.10.2. Si un problème est détecté dans la version 4.10.3, vous ne pouvez pas procéder à la mise à niveau vers cette version. Lorsqu'un correctif est disponible dans la version 4.10.4, vous pouvez mettre à jour votre cluster vers cette version.

Ce canal est adapté aux environnements de production, car les versions qu'il comporte sont testées par les ingénieurs Red Hat SRE et les services d'assistance.

## chapitre 14 | Description des mises à jour du cluster

- Le canal **fast - 4.10** peut fournir des mises à jour 4.10.1 et 4.10.2, mais pas 4.11.1. Ce canal est également pris en charge par Red Hat et peut être appliqué aux environnements de production.

Les administrateurs doivent spécifiquement choisir un canal de version mineure différent, tel que **fast - 4.11**, afin de procéder à une mise à niveau vers une nouvelle version dans une nouvelle version mineure.

- Le canal **candidate - 4.10** vous permet d'installer les dernières fonctions d'OpenShift. Par le biais de ce canal, vous pouvez effectuer une mise à niveau vers toutes les versions de *flux en z*, telles que 4.10.1, 4.10.2, 4.10.3, etc.

Vous utilisez ce canal pour accéder aux fonctions les plus récentes du produit dès qu'elles sont publiées. Ce canal convient aux environnements de pré-production et de développement.

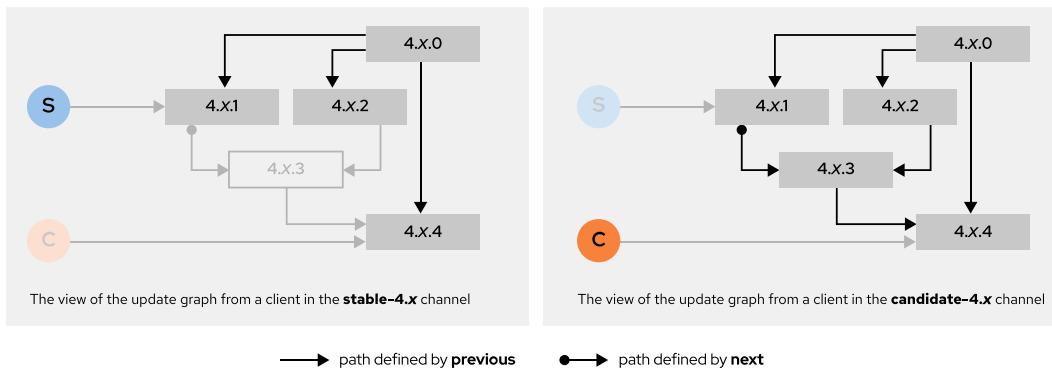
- Lorsque vous basculez vers le canal **eus - 4.10**, le canal **stable - 4.10** ne reçoit pas les mises à jour de *flux en z* tant que la prochaine version EUS n'est pas disponible.



### Note

À partir de la version 4.8 d'OpenShift Container Platform, Red Hat désignera toutes les versions mineures paires (par exemple, 4.8, 4.10, 4.12) comme étant des versions EUS (Extended Update Support).

Le graphique suivant décrit les graphiques de mise à jour pour les canaux stable et candidate :



**Figure 14.2: Graphiques de mise à jour pour les canaux stable et candidate**

Les canaux *stable* et *fast* sont classés en Disponibilité générale (GA), tandis que le canal *candidate* (canal release candidate) n'est pas pris en charge par Red Hat.

To ensure the stability of the cluster and the proper level of support, you should only switch from a *stable* channel to a *fast* channel, and vice versa. Although it is possible to switch from a *stable* channel or *fast* channel to a *candidate* channel, it is not recommended. The *candidate* channel is best suited for testing feature acceptance and assisting in qualifying the next version of OpenShift Container Platform.



### Note

La publication de mises à jour pour correctif et CVE peut prendre de plusieurs heures à une journée. Ce délai permet d'évaluer les impacts opérationnels sur les clusters OpenShift.

## Modification du canal de mise à jour

Vous pouvez modifier le canal de mise à jour pour le définir sur **stable-4.10**, **fast-4.10** ou **candidate-4.10** à l'aide de la console Web ou du client d'interface de ligne de commande OpenShift :

- Dans la console Web, accédez à la page **Administration > Cluster Settings** dans l'onglet **Details**, puis cliquez sur l'icône représentant un **pencil**.

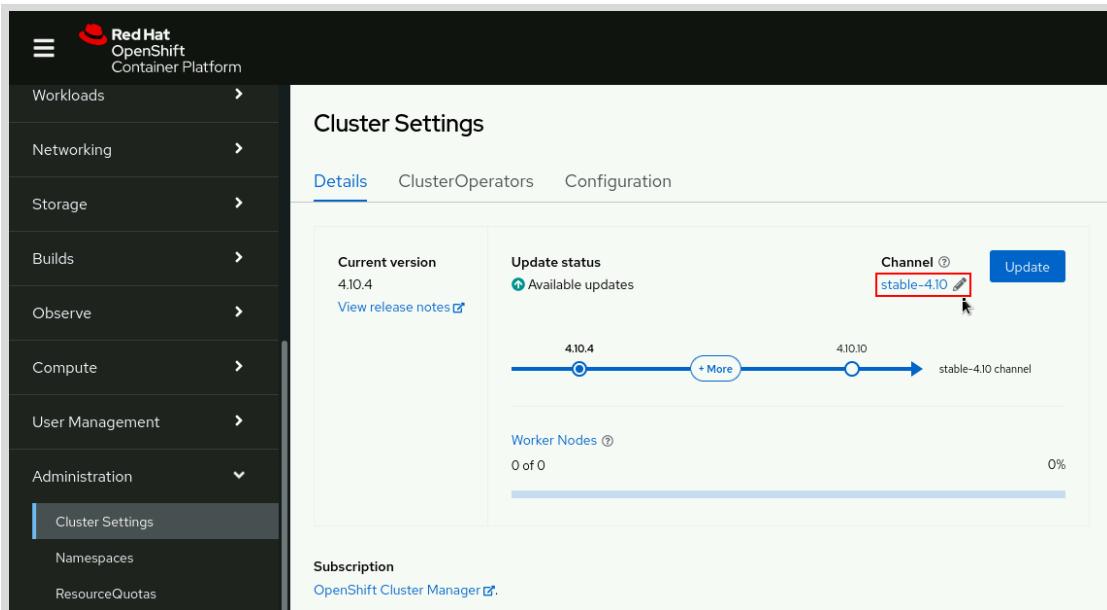


Figure 14.3: Canal de mise à jour actuel dans la console Web

Une fenêtre affiche les options permettant de sélectionner un canal de mise à jour.

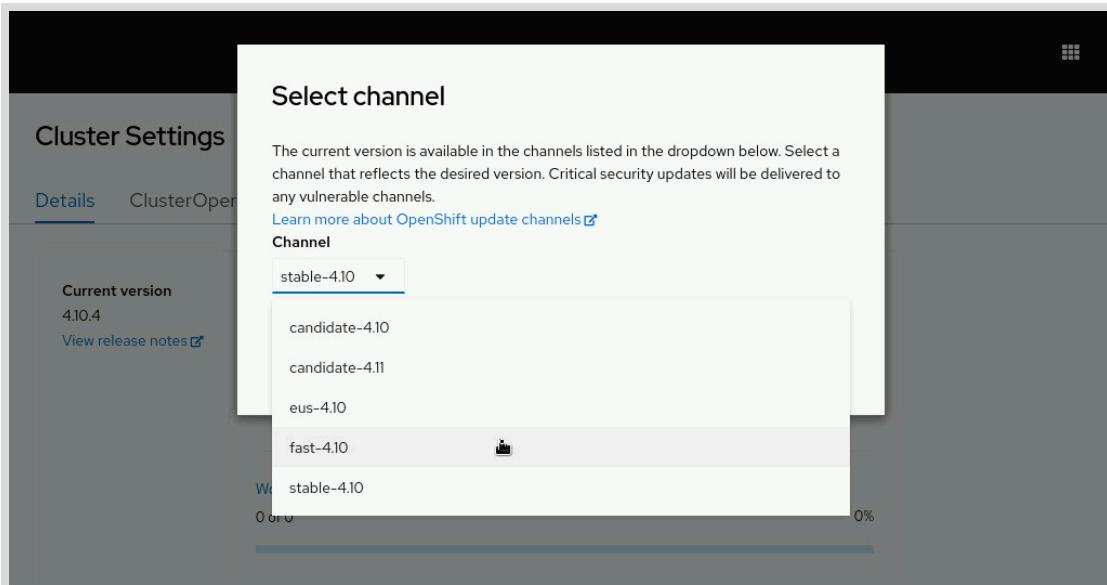


Figure 14.4: Modification du canal de mise à jour dans la console Web

- Exécutez la commande suivante pour basculer vers un autre canal de mise à jour à l'aide du client `oc`. Vous pouvez également basculer vers un autre canal de mise à jour, tel que `stable-4.10`, pour effectuer une mise à jour vers la version mineure suivante d'OpenShift Container Platform.

```
[user@host ~]$ oc patch clusterversion version --type="merge" \
>     --patch '[{"spec":{"channel":"fast-4.10"}}]'
clusterversion.config.openshift.io/version patched
```

## Description de l'OTA

L'OTA suit une approche client-serveur. Red Hat héberge les images de cluster et l'infrastructure de mise à jour. L'une des fonctionnalités de l'OTA est la génération de tous les scénarios de mise à jour possibles pour votre cluster. L'OTA rassemble des informations sur le cluster et votre droit à déterminer les scénarios de mise à niveau disponibles. La console Web envoie une notification lorsqu'une nouvelle mise à jour est disponible.

Le schéma suivant décrit l'architecture des mises à jour : Red Hat héberge à la fois les images de cluster et un « témoin » qui détecte automatiquement les nouvelles images transmises à Quay. L'opérateur de version de cluster (CVO) reçoit son état de mise à jour du témoin. Le CVO commence par mettre à jour les composants du cluster via leurs opérateurs, puis met à jour les composants supplémentaires gérés par Operator Lifecycle Manager (OLM).

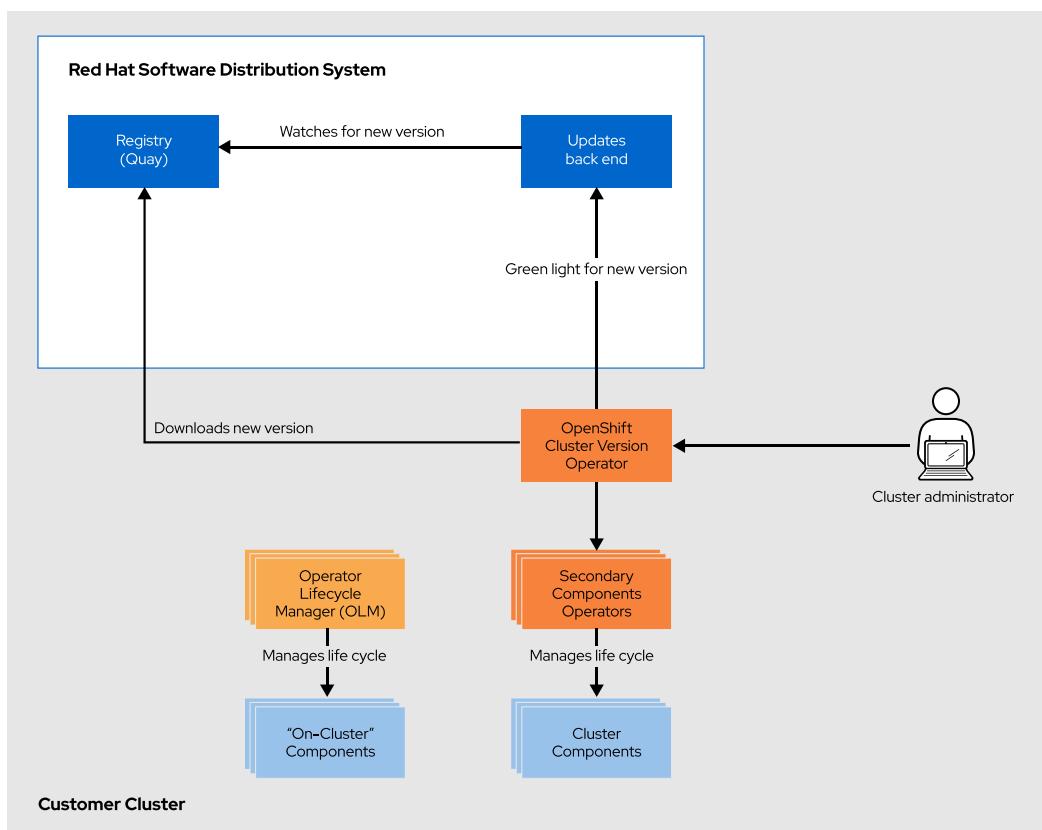


Figure 14.5: Architecture de mises à jour de la plateforme OpenShift Container Platform

La télémétrie permet à Red Hat de déterminer la procédure de mise à jour. Le cluster utilise la télémétrie basée sur Prometheus pour créer des rapports sur l'état de chaque opérateur de cluster. Les données sont rendues anonymes et renvoyées aux serveurs Red Hat qui informent les administrateurs de clusters des nouvelles versions potentielles.

**Note**

Les valeurs de Red Hat en matière de confidentialité des clients. Pour obtenir la liste complète des données collectées par Telemeter, consultez le document *Sample Metrics* listé dans la section Références.

À l'avenir, Red Hat a l'intention d'étendre la liste des opérateurs mis à jour compris dans la procédure de mise à niveau afin d'inclure des opérateurs d'éditeurs de logiciels indépendants (ISV).

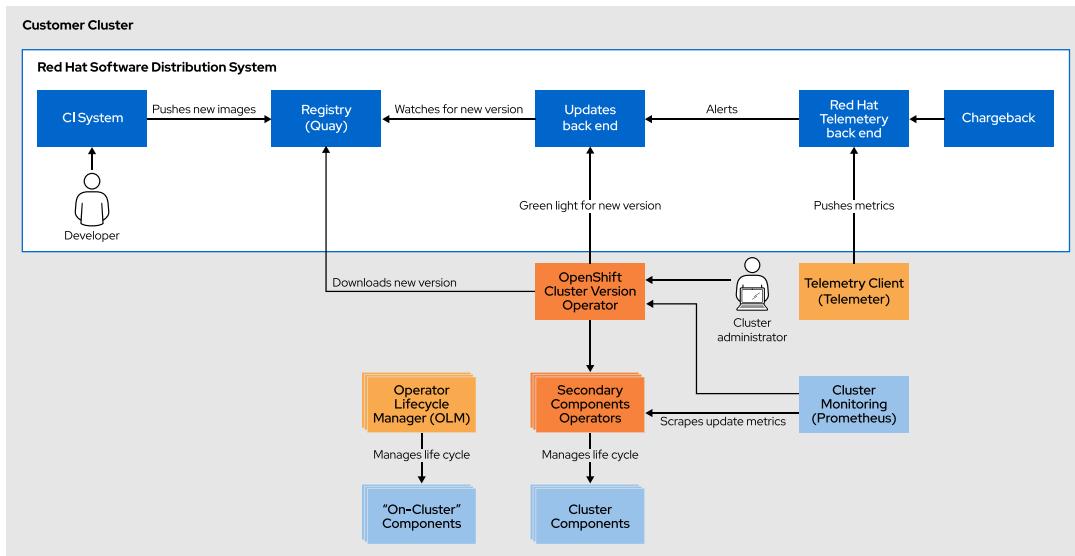


Figure 14.6: Gestion des mises à jour de cluster à l'aide de la télémétrie

## Présentation du processus de mise à jour

Deux composants sont impliqués dans le processus de mise à jour de cluster :

### Opérateur Machine Config

L'opérateur Machine Config applique l'état de machine souhaité à chacun des nœuds. Ce composant gère également la mise à niveau progressive des nœuds du cluster et le format de configuration utilisé est CoreOS Ignition.

### Gestionnaire de cycle de vie d'opérateur (OLM)

L'Operator Lifecycle Manager (OLM) orchestre les mises à jour de tous les opérateurs exécutés dans le cluster.

## Mise à jour du cluster

Vous pouvez mettre à jour le cluster via la console Web ou depuis la ligne de commande. La mise à jour depuis la console Web est plus simple que par la ligne de commande. La page **Administration > Cluster Settings** affiche le **statut** de *mises à jour disponibles* lorsqu'une nouvelle mise à jour est disponible. Depuis cette page, cliquez sur **Update now** pour débuter le processus :

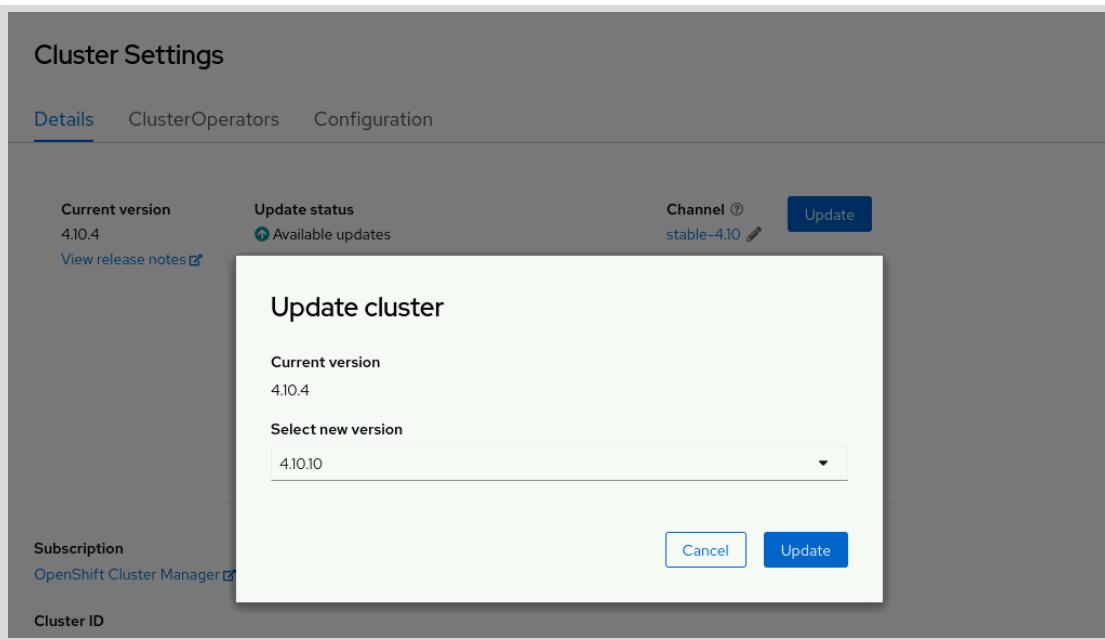


Figure 14.7: Mise à jour du cluster depuis la console Web

**Important**

Red Hat ne prend pas en charge le rétablissement, ou restauration, d'une version précédente de votre cluster. Red Hat ne prend en charge que la mise à niveau vers une version plus récente.

Le processus de mise à jour met également à jour le système d'exploitation sous-jacent lorsque des mises à jour sont disponibles. Il utilise la technologie `rpm-ostree` pour gérer les mises à niveau transactionnelles. Les mises à jour sont fournies via des images de conteneur et font partie du processus de mise à jour OpenShift. Lorsque la mise à jour est déployée, les nœuds extraient la nouvelle image, écrivent les paquets sur le disque, puis modifient le chargeur de démarrage pour démarrer dans la nouvelle version. La machine redémarre et implémente une mise à jour progressive afin de s'assurer que la capacité du cluster est affectée au minimum.

## Mettre à jour le cluster à l'aide de la ligne de commande

Les étapes suivantes décrivent la procédure de mise à jour d'un cluster en tant qu'administrateur de cluster à l'aide de l'interface de ligne de commande :

1. Veillez à mettre à jour tous les opérateurs installés via le gestionnaire de cycle de vie d'opérateur (OLM - Operator Lifecycle Manager) vers la dernière version avant de mettre à jour le cluster OpenShift.
2. Récupérez la version du cluster, examinez les informations actuelles du canal de mise à jour et confirmez le canal. Si vous exécutez le cluster en production, assurez-vous que le canal est stable.

```
[user@host ~]$ oc get clusterversion
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE    STATUS
version   4.10.3   True       False        43d      Cluster version is 4.10.3

[user@host ~]$ oc get clusterversion -o jsonpath='{.items[0].spec.channel}{"\n"}'
stable-4.10
```

3. Affichez les mises à jour disponibles et notez le numéro de version de la mise à jour que vous souhaitez appliquer.

```
[user@host ~]$ oc adm upgrade
Cluster version is 4.10.3

Updates:

VERSION IMAGE
4.10.4 quay.io/openshift-release-dev/ocp-release@sha256:...
...output omitted...
```

4. Appliquez la mise à jour la plus récente à votre cluster ou effectuez une mise à jour vers une version spécifique :

- Exécutez la commande suivante pour installer la dernière mise à jour disponible pour votre cluster.

```
[user@host ~]$ oc adm upgrade --to-latest=true
```

- Exécutez la commande suivante pour installer une version spécifique. **VERSION** correspond à l'une des versions disponibles que la commande `oc adm upgrade` renvoie.

```
[user@host ~]$ oc adm upgrade --to=VERSION
```



### Important

OpenShift Container Platform 4.9 utilise Kubernetes 1.22, qui a supprimé un nombre important d'API `v1beta1` obsolètes.

La version 4.8.14 d'OpenShift Container Platform a introduit une exigence selon laquelle un administrateur doit fournir un accusé de réception manuel avant que le cluster puisse être mis à niveau de la version 4.8 vers la version 4.9. Cela permet d'éviter les problèmes après la mise à niveau vers OpenShift Container Platform 4.9. En effet, les API qui ont été supprimées sont toujours utilisées par les charges de travail, les outils ou d'autres composants s'exécutant sur le cluster ou interagissant avec celui-ci.

Les administrateurs doivent évaluer leur cluster pour toutes les API en cours d'utilisation qui seront supprimées et migrer les composants affectés de sorte qu'ils utilisent la nouvelle version d'API appropriée. Une fois cette opération effectuée, l'administrateur peut fournir un accusé de réception de l'administrateur.

Tous les clusters nécessitent cet accusé de réception de l'administrateur avant de pouvoir être mis à niveau vers OpenShift Container Platform 4.9.

```
[user@host ~]$ oc patch configmap admin-acks -n openshift-config \
>   --type=merge \
>   --patch '{"data":{"ack-4.8-kube-1.22-api-removals-in-4.9":"true"}}'
configmap/admin-acks patched
```

5. La commande précédente initialise le processus de mise à jour. Exécutez la commande suivante pour vérifier l'état de l'opérateur de version de cluster (CVO - Cluster Version Operator) et les opérateurs de cluster installés.

```
[user@host ~]$ oc get clusterversion
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE    STATUS
version   4.10.3   True       True         30m      Working towards 4.10.4 ...

[user@host ~]$ oc get clusteroperators
NAME          VERSION  AVAILABLE  PROGRESSING  DEGRADED
authentication 4.10.3   True       False        False
cloud-credential 4.10.4   False      True        False
openshift-apiserver 4.10.4   True       False        True
...output omitted...
```

6. La commande suivante vous permet de consulter l'historique de l'état de la version du cluster pour surveiller l'état de la mise à jour. La mise à jour de tous les objets peut prendre un certain temps.

L'historique contient une liste des versions les plus récentes appliquées au cluster. Cette valeur est mise à jour lorsque le CVO applique une mise à jour. La liste est classée par date, où la mise à jour la plus récente est en première position dans la liste.

Si le déploiement a été effectué correctement, les mises à jour dans l'historique ont l'état `Completed`. Dans le cas contraire, la mise à jour a l'état `Partial` si elle a échoué ou n'est pas terminée.

```
[user@host ~]$ oc describe clusterversion
...output omitted...
History:
Completion Time: 2022-04-28T04:38:12Z
Image: quay.io/openshift-release-dev/ocp-release@sha256:...
Started Time: 2022-04-28T03:35:05Z
State: Partial
Verified: true
Version: 4.10.4
Completion Time: 2022-03-15T12:39:02Z
Image: quay.io/openshift-release-dev/ocp-release@sha256:...
Started Time: 2022-03-15T12:23:14Z
State: Completed
Verified: false
Version: 4.10.3
```



### Important

En cas d'échec d'une mise à niveau, l'opérateur s'interrompt et signale l'état du composant défectueux. Le fait de restaurer votre cluster vers une version antérieure n'est pas pris en charge.

**If your upgrade fails, contact Red Hat support.**

7. Une fois l'opération terminée, vous pouvez vérifier que le cluster a été mis à jour vers la nouvelle version.

```
[user@host ~]$ oc get clusterversion
NAME      VERSION  AVAILABLE  PROGRESSING  SINCE    STATUS
version   4.10.4  True       True        30m      Cluster version is 4.10.4
```



## Références

Pour plus d'informations sur l'installation de Red Hat OpenShift Container Platform dans un environnement déconnecté, reportez-vous au chapitre

*Installation configuration* de la documentation *Installing* de Red Hat

OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/installing/index#installation-configuration](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/installing/index#installation-configuration)

Pour plus d'informations sur les canaux de mise à jour, les conditions préalables à la mise à jour et la mise à jour de clusters dans des environnements déconnectés, reportez-vous aux chapitres *Updating a restricted network cluster* et *Updating a cluster between minor versions* de la documentation *Updating clusters* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/updating\\_clusters/index#updating-restricted-network-cluster](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/updating_clusters/index#updating-restricted-network-cluster)

Pour plus d'informations sur la mise à jour des opérateurs installés

via le gestionnaire de cycle de vie d'opérateur (OLM), reportez-vous à la section *Upgrading installed Operators* dans le chapitre

*Administrator tasks* de la documentation *Working with Operators* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/operators/index#olm-upgrading-operators](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/operators/index#olm-upgrading-operators)

Pour plus d'informations sur les graphiques de mise à jour

d'OpenShift Container Platform, consultez la page suivante sur le Portail Client :

[https://access.redhat.com/labs/ocpupgrade/graph/update\\_path](https://access.redhat.com/labs/ocpupgrade/graph/update_path)

Pour plus d'informations sur les API obsolètes dans

OpenShift Container Platform 4.9, consultez la page suivante sur le Portail Client :

<https://access.redhat.com/articles/6329921>

Pour plus d'informations sur la politique du cycle de vie

d'OpenShift Container Platform, consultez la page suivante sur le Portail Client :

<https://access.redhat.com/support/policy/updates/openshift>

Pour plus d'informations sur le canal de prise en charge prolongée des mises à jour (EUS) d'OpenShift, consultez la page suivante sur le Portail Client :

<https://access.redhat.com/support/policy/updates/openshift-eus>

Pour plus d'informations sur l'exécution d'une mise à jour EUS vers EUS, consultez la page suivante :

<https://docs.openshift.com/container-platform/4.10/updating/preparing-eus-eus-upgrade.html>

## ► Quiz

# Description du processus de mise à jour du cluster

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Quelles sont les deux mises à jour disponibles dans le canal de mise à jour fast -4.10 ? (Choisissez deux réponses.)**
  - a. 4.9.2
  - b. 4.10.1
  - c. 4.11.1
  - d. 4.10.5
- ▶ 2. **Parmi les composants suivants, lequel récupère les images de cluster mises à jour depuis Quay.io ?**
  - a. Surveillance de cluster (Prometheus)
  - b. Gestionnaire de cycle de vie d'opérateur (OLM)
  - c. Opérateur de version de cluster (CVO)
  - d. Client de télémétrie (Telemeter)
- ▶ 3. **Parmi les composants suivants, lequel gère les mises à jour des opérateurs qui ne sont pas des opérateurs de cluster ?**
  - a. Gestionnaire de cycle de vie d'opérateur (OLM)
  - b. Client de télémétrie (Telemeter)
  - c. Opérateur de version de cluster (CVO)
- ▶ 4. **Quelles sont les deux commandes qui vous permettent de récupérer la version du cluster en cours d'exécution ? (Choisissez deux réponses.)**
  - a. oc adm upgrade
  - b. oc get clusterchannel
  - c. oc get clusterversion

► 5. Parmi les propositions suivantes concernant la fonctionnalité OTA, lesquelles sont vraies ?

- a. Le canal *stable* est classé en Disponibilité générale (GA), tandis que le canal *fast* est classé en canal Release candidate (RC).
- b. Lorsque vous utilisez le canal stable, vous ne pouvez pas ignorer les versions intermédiaires. Par exemple, lors de la mise à jour de 4.9.27 vers 4.9.29, OpenShift installe d'abord la version 4.9.28.
- c. Il n'est pas recommandé de passer d'un canal stable ou d'un canal fast à un canal candidate. Cependant, vous pouvez passer d'un canal fast à un canal stable, et vice versa.
- d. La restauration d'une mise à jour ayant échoué est uniquement prise en charge par Red Hat lorsque vous tentez d'effectuer une mise à jour d'une version de flux en z vers une autre (par exemple, de 4.9.2 à 4.9.3, mais pas de 4.9.3 à 4.10).

► 6. Parmi les canaux suivants, deux sont classés comme disponibilité générale, lesquels ?  
(Choisissez deux réponses.)

- a. candidate-4.10.stable
- b. stable-4.10
- c. candidate-stable-4.10
- d. fast-4.10
- e. fast-4.10.1

## ► Solution

# Description du processus de mise à jour du cluster

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- ▶ 1. **Quelles sont les deux mises à jour disponibles dans le canal de mise à jour fast-4.10 ? (Choisissez deux réponses.)**
  - a. 4.9.2
  - b. 4.10.1
  - c. 4.11.1
  - d. 4.10.5
- ▶ 2. **Parmi les composants suivants, lequel récupère les images de cluster mises à jour depuis Quay.io ?**
  - a. Surveillance de cluster (Prometheus)
  - b. Gestionnaire de cycle de vie d'opérateur (OLM)
  - c. Opérateur de version de cluster (CVO)
  - d. Client de télémétrie (Telemeter)
- ▶ 3. **Parmi les composants suivants, lequel gère les mises à jour des opérateurs qui ne sont pas des opérateurs de cluster ?**
  - a. Gestionnaire de cycle de vie d'opérateur (OLM)
  - b. Client de télémétrie (Telemeter)
  - c. Opérateur de version de cluster (CVO)
- ▶ 4. **Quelles sont les deux commandes qui vous permettent de récupérer la version du cluster en cours d'exécution ? (Choisissez deux réponses.)**
  - a. oc adm upgrade
  - b. oc get clusterchannel
  - c. oc get clusterversion

► 5. Parmi les propositions suivantes concernant la fonctionnalité OTA, lesquelles sont vraies ?

- a. Le canal *stable* est classé en Disponibilité générale (GA), tandis que le canal *fast* est classé en canal Release candidate (RC).
- b. Lorsque vous utilisez le canal stable, vous ne pouvez pas ignorer les versions intermédiaires. Par exemple, lors de la mise à jour de 4.9.27 vers 4.9.29, OpenShift installe d'abord la version 4.9.28.
- c. Il n'est pas recommandé de passer d'un canal stable ou d'un canal fast à un canal candidate. Cependant, vous pouvez passer d'un canal fast à un canal stable, et vice versa.
- d. La restauration d'une mise à jour ayant échoué est uniquement prise en charge par Red Hat lorsque vous tentez d'effectuer une mise à jour d'une version de flux en z vers une autre (par exemple, de 4.9.2 à 4.9.3, mais pas de 4.9.3 à 4.10).

► 6. Parmi les canaux suivants, deux sont classés comme disponibilité générale, lesquels ?  
(Choisissez deux réponses.)

- a. candidate-4.10.stable
- b. stable-4.10
- c. candidate-stable-4.10
- d. fast-4.10
- e. fast-4.10.1

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- L'un des principaux avantages des modifications de l'architecture OpenShift 4 est que vous pouvez mettre à jour vos clusters Over-the-Air (OTA).
- Red Hat offre un nouveau système de distribution de logiciels qui fournit la meilleure procédure de mise à jour de votre cluster et du système d'exploitation sous-jacent.
- Il existe plusieurs canaux de distribution :
  - Le canal *stable* propose des mises à jour retardées.
  - Le canal *fast* propose les mises à jour dès qu'elles sont disponibles.
  - Le canal *candidate* fournit des mises à jour pour tester l'acceptation des fonctions dans la prochaine version d'OpenShift Container Platform.
  - Le canal *eus* (disponible uniquement lorsque la version 4.10 est exécutée) étend la phase de maintenance pour les clients disposant d'abonnements Premium.
- Red Hat ne prend pas en charge le rétablissement d'une version précédente de votre cluster. Red Hat ne prend en charge que la mise à niveau vers une version plus récente.

## chapitre 15

# Gestion d'un cluster avec la console Web

### Objectif

Gestion d'un cluster Red Hat OpenShift à l'aide de la console Web.

### Résultats

- Exécuter l'administration du cluster à l'aide de la console Web.
- Gérer des applications et des opérateurs Kubernetes à l'aide de la console Web.
- Examiner les mesures de performances et d'intégrité pour des applications et des nœuds de cluster.

### Sections

- Exécution de l'administration du cluster (et exercice guidé)
- Gestion des charges de travail et des opérateurs (et exercice guidé)
- Examen des mesures du cluster (et exercice guidé)

### Atelier

Gestion d'un cluster avec la console Web

# Exécution de l'administration du cluster

## Résultats

Après avoir terminé cette section, vous serez en mesure d'exécuter l'administration du cluster avec la console Web.

## Description de la console Web

La console Web Red Hat OpenShift offre une interface utilisateur graphique qui permet d'effectuer des tâches d'administration, de gestion et de dépannage. Elle prend en charge les perspectives **Administrator** et **Developer**. Ce cours explore la perspective **Administrator**.

La liste suivante présente certains des composants les plus importants de la console Web, regroupés en fonction des éléments du menu de navigation principal. La possibilité d'afficher des éléments particuliers dépend des rôles et des liaisons de rôle associés à un utilisateur. Les utilisateurs disposant du rôle de cluster `cluster-admin` peuvent tout afficher et tout modifier. Les utilisateurs disposant du rôle de cluster `view` peuvent afficher la plupart des éléments, mais ils ne peuvent pas y apporter de modifications. Des rôles supplémentaires peuvent permettre l'accès à des éléments spécifiques.

### Home

La page **Home > Overview** fournit un aperçu rapide du cluster, notamment les métriques d'intégrité, le nombre de ressources et une liste de diffusion des événements, tels que les mises à jour machine ou les défaillances de pod.

Vous pouvez accéder à la page **Home > Search** pour trouver ou créer des ressources de tout type. C'est aussi un point de départ utile pour accéder à des ressources n'ayant pas d'accès dédié dans le menu.

La page **Home > Events** affiche un flux d'événements filtrables qui se produisent dans le cluster. C'est un bon point de départ pour la résolution de problèmes.

### Opérators

Explore and install operators curated by Red Hat using OperatorHub, then navigate to the **Installed Operators** page to manage the operators.

### Charges de travail, mise en réseau et stockage

Gérer les ressources communes telles que les déploiements, les services et les volumes persistants. La capacité à afficher les journaux de pod et à se connecter à un terminal peut être utilisée pour la résolution de problèmes.

### Builds

Gérer les configurations de compilation, les compilations et les flux d'images.

### Surveillance

Afficher les alertes et effectuer des requêtes Prometheus ad hoc.

### Compute

Afficher et gérer les ressources de calcul telles que les nœuds, les machines et les mises à l'échelle automatique de machines.

## Gestion des utilisateurs

Afficher et gérer les utilisateurs, les groupes, les comptes de service, les rôles et les liaisons de rôles.

## Administration

Afficher et gérer un large éventail de paramètres présentant un intérêt pour les administrateurs de cluster, tels que les mises à jour de cluster, les opérateurs de cluster, les CRD et les quotas de ressources.

# Accès à la console Web OpenShift

La console Web OpenShift est exécutée en tant que pod dans le projet `openshift-console` et est gérée par l'opérateur exécuté dans le projet `openshift-console-operator`. Vous pouvez découvrir l'URL de la console en utilisant la commande `oc whoami`.

```
[user@host ~]$ oc whoami --show-console  
https://console-openshift-console.apps.cluster.example.com
```

Dans les systèmes autres que de production, les certificats auto-signés sont couramment utilisés pour le point d'accès HTTPS. Les navigateurs Web vous mettront en garde contre le certificat, et vous devrez ajouter une exception de sécurité lorsque vous accédez à la console Web pour la première fois.

## Trouver les ressources

L'interface utilisateur Web offre plusieurs moyens de localiser les ressources. De nombreuses ressources communes, telles que les `deployments` et `services`, sont disponibles dans le menu principal situé à gauche. Vous pouvez utiliser la page `Home > Search` pour trouver d'autres types de ressources. Cette page fournit un menu complet des types de ressources et un champ de sélecteur d'étiquette.

Utilisez le filtre par nom pour localiser rapidement des ressources sur des pages contenant de longues listes comme la page **Projects** :

Name	Display name	Status	Requester	Memory	CPU	Created
openshift-apiserver	No display name	Active	No requester	659.2 MiB	0.065 cores	Mar 15, 2022, 8:25 AM
openshift-apiserver-operator	No display name	Active	No requester	98.8 MiB	0.007 cores	Mar 15, 2022, 8:23 AM
openshift-kube-apiserver	No display name	Active	No requester	4,941.9 MiB	0.379 cores	Mar 15, 2022, 8:23 AM

Il peut être utile de filtrer les pods par état pour identifier les problèmes potentiels ou les déploiements problématiques :

Status	Ready	Restarts	Owner	Memory	CPU	Created
<input type="checkbox"/> Running 3	2/2	2	RS apiserver-76db4c4d6d	218.4 MiB	0.022 cores	Apr 28, 2022, 12:19 AM
<input type="checkbox"/> Pending 0						
<input type="checkbox"/> Terminating 0						
<input type="checkbox"/> CrashLoopBackOff 0						
<input type="checkbox"/> Completed 0						
<input type="checkbox"/> Failed 0						
<input type="checkbox"/> Unknown 0						

La page de détails d'une ressource affiche des informations communes utiles. Le contenu de cette page varie en fonction des différents types. Par exemple, la page **Pod Details** affiche les mesures et les informations d'état, et la page **Secret Details** vous permet de révéler ou de copier des données stockées dans le secret. Les pages de détails fournissent un éditeur YAML qui permet d'afficher et de modifier la spécification de ressources depuis la console Web. Certains types de ressources, tels que les **secrets** et les **role bindings**, offrent des interfaces utilisateur plus avancées adaptées au type de ressource.

## Création d'utilisateurs et de groupes

La page **User Management > Users** affiche les utilisateurs qui se sont préalablement connectés à OpenShift. Comme indiqué dans *Chapitre 10, Configuration de l'authentification et de l'autorisation*, OpenShift prend en charge plusieurs fournisseurs d'identité (IdP), notamment HTPasswd, LDAP et OpenID Connect.

Lors de l'utilisation du fournisseur d'identité HTPasswd, l'éditeur de **secrets** peut simplifier l'ajout, la mise à jour ou la suppression d'entrées dans le secret HTPasswd. Après avoir utilisé un terminal pour générer une entrée HTPasswd nouvelle ou mise à jour, basculez vers la console Web pour modifier le secret.

Dans l'interface utilisateur Web, localisez le secret dans le projet **openshift-config**, puis cliquez sur **Actions > Edit Secret**. L'outil **Edit Key/Value Secret** gère l'encodage en base64 pour vous. Ajoutez une ligne pour permettre à un nouvel utilisateur de se connecter à OpenShift. Mettez à jour une ligne pour changer le mot de passe d'un utilisateur. Supprimer une ligne afin qu'un utilisateur ne puisse pas se connecter à OpenShift.

La page **User Management > Groups** affiche les groupes existants et permet de créer des groupes.

## Création d'un projet

L'interface utilisateur Web présente une multitude de pages et de formulaires pour la configuration de projets. Pour créer un projet :

- Accédez à la page **Home > Projects** pour afficher la liste complète des projets. Cliquez ensuite sur **Create Project** et renseignez le formulaire pour créer un projet.
- Une fois que vous avez créé votre nouveau projet, vous pouvez accéder à l'onglet **Role Bindings** de la page **Project Details**.

- Red Hat recommande aux administrateurs chargés des clusters multitenants de configurer des `resource quotas` et des `limit ranges`, qui appliquent respectivement les limites totales du projet et les limites de conteneur. Accédez à **Administration > Resource Quotas** ou **Administration > Limit Ranges** pour accéder à l'éditeur YAML approprié, où vous pouvez configurer ces limites.

## À propos des limites

La console Web OpenShift est un outil puissant pour l'administration graphique des clusters OpenShift. Toutefois, certaines tâches d'administration ne sont pas actuellement disponibles dans la console Web. Par exemple, l'affichage des journaux de nœuds et l'exécution de sessions de débogage de nœuds nécessitent l'outil de ligne de commande `oc`.



### Références

Pour plus d'informations, reportez-vous à la documentation *Web console* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse  
[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/web\\_console/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/web_console/index)

## ► Exercice guidé

# Exécution de l'administration du cluster

Dans cet exercice, vous allez exécuter l'administration du cluster à l'aide de la console Web.

## Résultats

Vous devez pouvoir utiliser la console Web OpenShift pour :

- Trouver des ressources associées à un opérateur.
- Passer en revue l'état d'un pod, la définition YAML et les journaux.
- Afficher et modifier les ressources de configuration du cluster.
- Créer un nouveau projet et configurer ses quotas de ressources, plages de limites et contrôles d'accès basés sur les rôles (RBAC).

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée les ressources nécessaires à cet exercice.

```
[student@workstation ~]$ lab console-admin start
```

## Instructions

- 1. En tant qu'utilisateur `admin`, localisez la console Web OpenShift et accédez-y.
- 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Identifiez l'URL de la console Web.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Ouvrez un navigateur Web et accédez à l'URL de la console Web :
  - <https://console-openshift-console.apps.ocp4.example.com>
- 1.4. Cliquez sur `localusers` et connectez-vous en tant qu'utilisateur `admin` avec le mot de passe `redhat`.

- 2. Examinez les journaux de pod `openshift-console-operator` et `openshift-console`.
- 2.1. Dans l'interface utilisateur Web de Red Hat OpenShift Container Platform, cliquez sur `Home > Projects` pour afficher la page `Projects`.
  - 2.2. Saisissez `console` dans le champ `Search by name`, puis cliquez sur le lien `openshift-console-operator`.

Name	Display name	Status	Requester	Memory	CPU	Created
openshift-console	No display name	Active	No requester	134.4 MiB	0.007 cores	May 3, 2022, 7:05 AM
openshift-console-operator	No display name	Active	No requester	46.8 MiB	0.006 cores	May 3, 2022, 7:05 AM
openshift-console-user-settings	No display name	Active	No requester	-	-	May 3, 2022, 7:05 AM

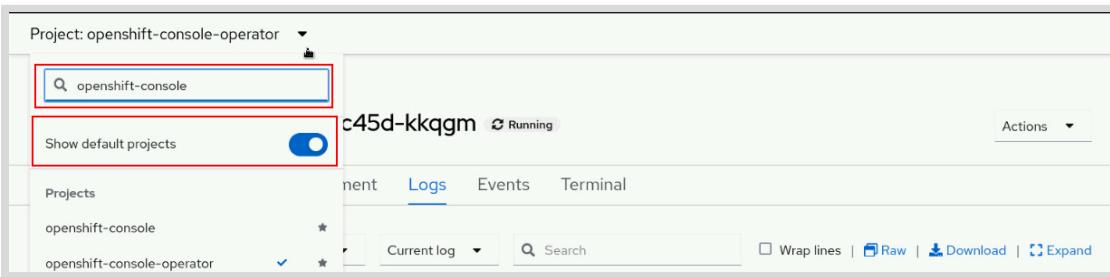
- 2.3. Cliquez sur `Workloads`, puis cliquez sur `1 of 1 Pod` pour accéder à l'ensemble de répliques de `console-operator`.

Deployment		52.8 MiB	0.005 cores	1 of 1 Pod
console-operator				

- 2.4. Cliquez sur le nom du pod marqué par l'icône P pour naviguer jusqu'au pod `console-operator`.

Name	Status	Ready	Restarts	Node	Memory	CPU	Created
console-operator-6999ddc45d-kkqgm	Running	1/1	4	master03	49.8 MiB	0.006 cores	May 3, 2022, 7:22 AM

- 2.5. Examinez la page **Pod Details** et notez les métriques des pods, l'état d'exécution et les volumes.
- 2.6. Cliquez sur **YAML** pour accéder à l'éditeur de ressources de pods.
- 2.7. Cliquez sur **Logs** pour afficher les journaux de l'opérateur de la console.
- 2.8. Ouvrez la liste **Project** et saisissez `openshift-console` pour basculer vers le projet `openshift-console`. Cliquez sur l'option **Show default projects** si elle est désactivée.



- 2.9. Cliquez sur le premier pod du tableau, puis cliquez sur **Logs** pour afficher les journaux de pods de la console.
- 3. Passez en revue les paramètres Console, Image et OAuth cluster.
- 3.1. Cliquez sur **Administration > Cluster Settings** dans le menu de gauche pour afficher la page **Cluster Settings**. Notez que les informations sur le canal de mise à jour et la version actuelle du cluster sont répertoriées en haut et qu'une section pour l'historique des mises à jour du cluster est présentée ci-dessous.
  - 3.2. Cliquez sur **Configuration** pour accéder à la liste des ressources de configuration de cluster.
  - 3.3. Cliquez sur **Console** (`config.openshift.io`), puis sur **YAML** pour passer en revue la ressource **Console**.
  - 3.4. Revenez à la page de configuration **Cluster Settings**. Cliquez sur **Image**, puis sur **YAML**. Notez que `internalRegistryHostname` est configuré pour utiliser le registre d'images interne.
  - 3.5. Revenez à la page de configuration **Cluster Settings** et cliquez sur **OAuth**. La page de présentation **OAuth Details** dispose d'une section spéciale pour répertorier et ajouter des fournisseurs d'identité. Accédez à la page **YAML** pour afficher des informations supplémentaires sur la configuration.
- 4. Passez en revue les rôles de cluster `admin`, `edit` et `view`.
- 4.1. Cliquez sur **User Management > Roles** dans le menu de gauche pour afficher la page **Roles**.
  - 4.2. Cliquez sur **admin** à côté de l'icône CR. Examinez le tableau **Rules** qui décrit les actions autorisées pour diverses ressources.

The screenshot shows a table of Cluster Roles. The columns are 'Name' and 'Namespace'. There are three rows:

- CR admin**: Namespace: All Namespaces
- CR aggregate-olm-edit**: Namespace: All Namespaces
- CR aggregate-olm-view**: Namespace: All Namespaces

- 4.3. Revenez à la page **Cluster Roles** et cliquez sur le rôle de cluster nommé **edit** pour afficher les détails du rôle de cluster **edit**.
- 4.4. Revenez à la page **Cluster Roles** et tapez **view** dans le champ **Search by name**. Cliquez sur le rôle de cluster nommé **view** pour accéder aux détails du rôle de cluster **view**. Notez que ce rôle autorise uniquement les actions **get**, **list** et **watch** sur les ressources répertoriées.
- ▶ 5. Ajoutez une entrée d'utilisateur **tester** au secret **localusers**.
  - 5.1. Cliquez sur **Workloads > Secrets** dans le menu de gauche, puis sélectionnez **openshift-config** dans la liste de filtres **Project** pour afficher les secrets du projet **openshift-config**. Cliquez sur l'option **Show default projects** si elle est désactivée.
  - 5.2. Utilisez le filtre ou faites défiler la page vers le bas pour le localiser, puis cliquez sur le lien **localusers**.
  - 5.3. Cliquez sur **Actions > Edit Secret** pour naviguer jusqu'à l'outil **Edit Key/Value Secret**.
  - 5.4. Utilisez le terminal **workstation** pour générer une entrée **htpasswd** à l'aide du mot de passe **redhat**.

```
[student@workstation ~]$ htpasswd -n -b tester redhat
tester:$apr1$oQ3BtWOp.HtW97.$wVbJBofBNsNd4sd
```

- 5.5. Ajoutez la sortie de terminal de la commande **htpasswd** à la valeur **htpasswd** dans l'éditeur de secrets de la console Web OpenShift, puis cliquez sur **Save**.

```
admin:$apr1$Au9.fFr$0k5wvUBd3eeBt0baa77.dae
leader:$apr1$/abo4Hybn7a.tG5Zo0Bn.QWefXckiy1
developer:$apr1$RjqTY4cv$xql3.BQfg42moSxwnTNkh.
tester:$apr1$oQ3BtWOp.HtW97.$wVbJBofBNsNd4sd
```

- ▶ 6. Créez et configurez un nouveau projet nommé **console-apps**.
  - 6.1. Cliquez sur **Home > Projects** dans le menu de gauche pour afficher la page **Projects**, puis sur **Create Project**.
  - 6.2. Utilisez les informations suivantes pour le nouveau projet, puis cliquez sur **Create**.

## Créer un formulaire de projet

Champ	Valeur
Name	console-apps
Display Name	Applications de chapitre de console
Description	Exemple de projet

- 6.3. Cliquez sur **Administration > Resource Quotas** dans le menu de gauche, puis cliquez sur **Create Resource Quota**. Modifiez le document YAML comme suit et cliquez ensuite sur **Create**.

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: quota
  namespace: console-apps
spec:
  hard:
    pods: '10'
    requests.cpu: '2'
    requests.memory: 8Gi
    limits.cpu: '4'
    limits.memory: 12Gi
```

- 6.4. Cliquez sur **Administration > Limit Ranges** dans le menu de gauche, puis cliquez sur **Create Limit Range**. Modifiez le document YAML pour spécifier un nom pour la plage de limites. Indiquez les demandes et les limites de mémoire et de processeur par défaut du conteneur, puis cliquez sur **Create**.

```
apiVersion: v1
kind: LimitRange
metadata:
  name: limit-range
  namespace: console-apps
spec:
  limits:
  - default:
      cpu: 500m
      memory: 5Gi
    defaultRequest:
      cpu: 10m
      memory: 100Mi
    type: Container
```

- 6.5. Cliquez sur **User Management > Groups** dans le menu de gauche, puis sur **Create Group**. Utilisez l'éditeur pour définir une ressource Group comme suit, puis cliquez sur **Create**.

```

apiVersion: user.openshift.io/v1
kind: Group
metadata:
  name: project-team
users:
  - developer
  - tester

```

- 6.6. Cliquez sur **User Management > Role Bindings** dans le menu de gauche, puis cliquez sur **Create Binding**. Complétez le formulaire comme suit afin de créer une liaison de rôle pour le groupe **project-team**.

#### Formulaire Team Role Binding

Champ	Valeur
Binding Type	Liaison de rôle d'espace de noms (RoleBinding)
Name	team
Namespace	console-apps
Role Name	edit
Subject	Group
Subject Name	project-team

Cliquez sur **Create** pour créer le RoleBinding dans l'espace de noms **console-apps**.

- 6.7. Revenez à la page **Role Bindings** et cliquez sur **Create Binding** pour créer une liaison de rôle pour l'utilisateur **leader**. Remplissez le formulaire comme suit :

#### Formulaire Leader Role Binding

Champ	Valeur
Binding Type	Liaison de rôle d'espace de noms (RoleBinding)
Name	leader
Namespace	console-apps
Role Name	admin
Subject	User
Subject Name	leader

Cliquez sur **Create** pour créer le RoleBinding dans l'espace de noms **console-apps**.

- 6.8. Cliquez sur **admin > Log out**, puis reconnectez-vous en tant qu'utilisateur **developer** avec le mot de passe **developer**.

Assurez-vous que le compte du **developer** peut uniquement voir le projet **console-apps**.



**Note**

Les projets des exercices guidés précédents non supprimés au terme de leur exécution peuvent également s'afficher dans la liste.

- 6.9. Vous allez continuer à utiliser le nouveau projet **console-apps** dans la section suivante, il n'est donc pas nécessaire de le supprimer.

## Fin

Sur la machine **workstation**, utilisez la commande **lab** pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab console-admin finish
```



**Important**

Ne supprimez pas le projet **console-apps**. Il sera utilisé dans les sections suivantes.

L'exercice guidé est maintenant terminé.

# Gestion des charges de travail et des opérateurs

---

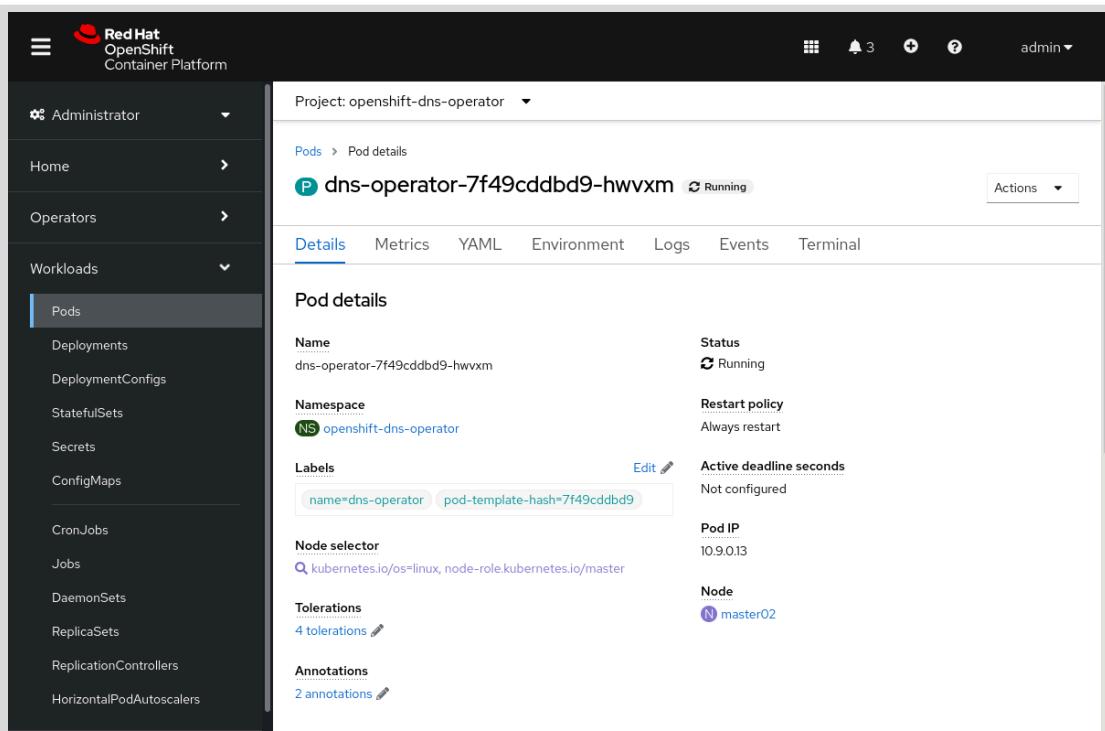
## Résultats

Après avoir terminé cette section, vous devez pouvoir gérer les applications et les opérateurs Kubernetes à l'aide de la console Web.

## Exploration des ressources de charge de travail

Les ressources de la charge de travail, telles que Pods, Deployments, Stateful Sets, et Config Maps sont répertoriées dans le menu **Workloads**. Cliquez sur un type de ressource pour afficher une liste de ressources, puis cliquez sur le nom de la ressource pour accéder à la page de détails de cette ressource.

Par exemple, pour naviguer jusqu'au pod d'opérateur DNS OpenShift, cliquez sur **Workloads > Pods**, sur **Show default projects**, sélectionnez **openshift-dns-operator** dans la liste Project située en haut de la page, puis cliquez sur le nom du pod répertorié dans le tableau :



The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is collapsed. The main navigation bar at the top has the Red Hat logo and the text "Red Hat OpenShift Container Platform". On the far right, there are icons for notifications (3), a plus sign, a question mark, and a dropdown for "admin". The main content area has a header "Project: openshift-dns-operator" and a breadcrumb "Pods > Pod details". Below this, a pod named "dns-operator-7f49cddb9-hwvxm" is listed as "Running". A "Actions" dropdown is shown next to it. The main panel is titled "Pod details" and contains the following information:

- Name:** dns-operator-7f49cddb9-hwvxm    **Status:** Running
- Namespace:** openshift-dns-operator    **Restart policy:** Always restart
- Labels:** name=dns-operator, pod-template-hash=7f49cddb9 (with an "Edit" button)
- Active deadline seconds:** Not configured
- Node selector:** kubernetes.io/os=linux, node-role.kubernetes.io/master
- Tolerations:** 4 tolerations
- Annotations:** 2 annotations
- Pod IP:** 10.9.0.13
- Node:** master02

Il existe souvent plusieurs façons de naviguer vers des ressources communes. Dans toute l'interface utilisateur Web, les ressources associées sont souvent liées les unes aux autres. La page Deployment Details affiche une liste de pods. Cliquez sur le nom d'un pod dans cette liste pour afficher la page Pod Details pour ce pod.

## Gestion des charges de travail

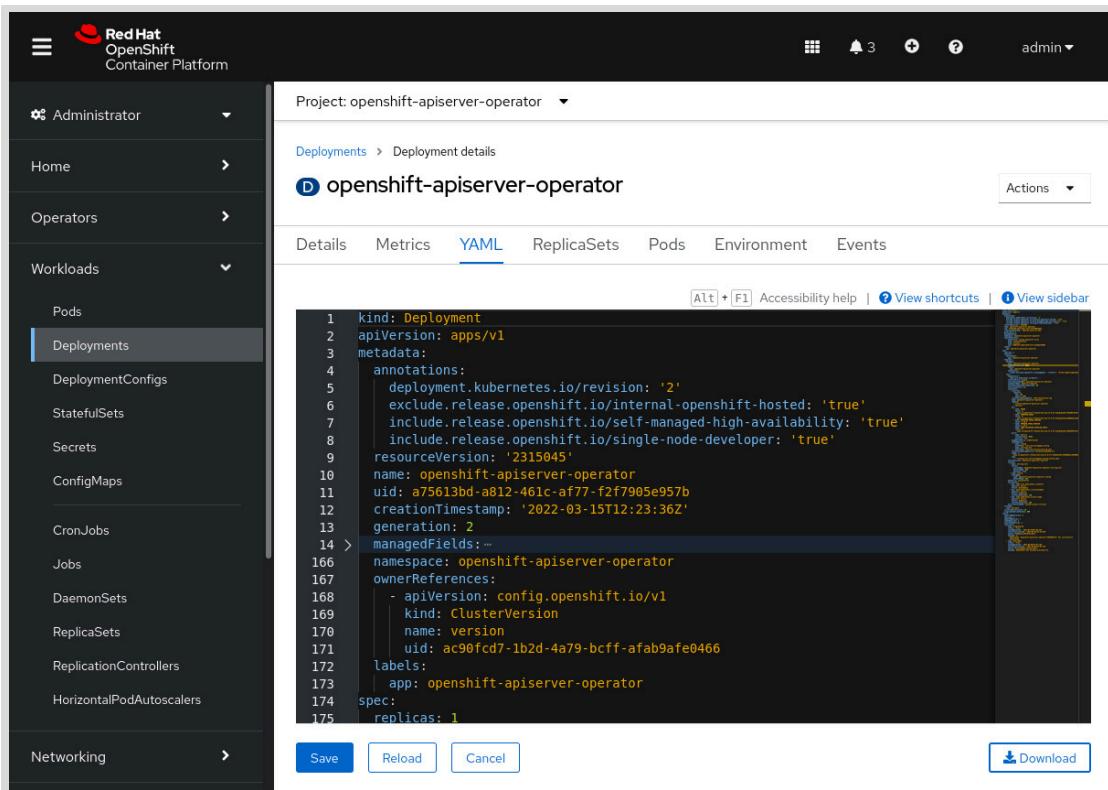
La console Web fournit des pages d'éditeur spécialisées pour de nombreuses ressources de charge de travail. Utilisez le menu **Actions** de la page de détails de la ressource pour accéder aux pages d'éditeur spécialisées :

Figure 15.9: Utilisation du menu Actions pour modifier un déploiement

Certaines pages d'action utiles sont décrites ci-dessous :

- Toutes les ressources comportent des éditeurs **Edit Labels** et **Edit Annotations**.
- Cliquez sur **Actions > Add Storage** pour ajouter une Revendication de volume persistant (PVC) à un déploiement.
- Pour modifier le nombre de répliques, accédez à la page **Deployment Details**, puis cliquez sur **Actions > Edit Pod Count**.
- Pour modifier la stratégie de mise à jour pour un déploiement, comme les paramètres de mise à jour progressive, accédez à la page **Deployment Details**, puis cliquez sur **Actions > Edit Update Strategy**. Pour modifier la stratégie de mise à jour d'une configuration de déploiement, accédez à la page **Deployment Config Details** puis cliquez sur **Actions > Edit Deployment Config**
- Accédez à la page **Secret Details** et cliquez sur **Actions > Edit Secret** pour afficher l'outil **Edit Key/Value Secret** qui utilise automatiquement Base64 pour coder et décoder les valeurs.

Vous pouvez également utiliser l'éditeur YAML intégré pour créer ou modifier des ressources de charge de travail. Glissez et déposez un fichier JSON ou YAML dans l'éditeur sur navigateur pour mettre à jour la ressource à partir d'un fichier sans utiliser la commande oc :



**Figure 15.10: Modification d'une ressource à l'aide de l'éditeur YAML intégré**

Outre la possibilité de modifier des ressources dans une page dédiée ou dans l'éditeur YAML intégré, vous pouvez effectuer de nombreuses autres opérations courantes directement depuis la console Web OpenShift. Par exemple, pour supprimer une ressource, accédez à la page de détails de la ressource, puis cliquez sur **Actions > Delete Resource Type**.

Il existe souvent plusieurs moyens d'effectuer une tâche donnée. Par exemple, pour mettre à l'échelle manuellement un déploiement, vous pouvez accéder à la page **Deployment Details**, puis cliquer sur **Actions > Edit Pod Count**. Vous pouvez également cliquer sur les flèches à côté du nombre de pods sans quitter la page.

## Déploiement d'applications

Vous pouvez créer des ressources de déploiement à partir de la page **Workloads > Deployments**. Cette section fournit un éditeur YAML doté d'une spécification préremplie pour définir votre ressource YAML.

La section **Builds** contient des outils pour :

- Créer des configurations de compilation pour les compilations Source-to-Image (S2I), Dockerfile ou personnalisées.
- Lister et inspecter les compilations.
- Gérer les flux d'images.

Après avoir lancé un déploiement ou une compilation, utilisez les pages de détails et d'événements de la ressource pour vérifier la réussite de l'opération ou commencez à rechercher la cause d'un échec de déploiement.

## Installation et utilisation d'opérateurs

Explorez les opérateurs de communauté et de partenaire sur la page Operators > OperatorHub de la console Web OpenShift. Bon nombre d'opérateurs peuvent être installés depuis la console Web. Cela inclut les opérateurs communautaires, non pris en charge par Red Hat.

Les opérateurs ajoutent des fonctionnalités et services à votre cluster, ainsi que l'automatisation traditionnellement effectuée par les opérateurs humains, comme la coordination de déploiement ou les sauvegardes automatiques. Les opérateurs couvrent un large éventail de catégories, notamment :

- Les bases de données traditionnelles telles que PostgreSQL et MySQL.
- Des structures big data populaires, telles qu'Apache Spark.
- Les plateformes de diffusion en continu basées sur Kafka, telles que les flux Red Hat AMQ.
- La structure sans serveur Knative, OpenShift Serverless Operator.

Cliquez sur la liste Operator pour afficher des informations détaillées sur l'opérateur, telles que sa version et où trouver la documentation. Lorsque vous êtes prêt à installer un opérateur, cliquez sur **Install** pour démarrer l'installation. Remplissez le formulaire **Operator Installation** pour sélectionner l'espace de noms et la stratégie d'approbation de l'opérateur cibles. Vous pouvez installer des opérateurs pour cibler tous les espaces de noms ou seulement certains. Sachez toutefois que tous les opérateurs ne prennent pas en charge toutes les options d'installation cible.

Une fois que vous avez installé un opérateur, il apparaît sur la page Operators > Installed Operators. S'il est installé pour un espace de noms spécifique, assurez-vous de sélectionner le bon projet à l'aide du filtre de projet situé en haut de la page :

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is a navigation menu with sections like Home, Operators, OperatorHub, Workloads, Networking, Storage, Builds, Observe, and Compute. The 'Operators' section is expanded, and 'OperatorHub' is selected. Under 'OperatorHub', 'Installed Operators' is selected. The main content area shows the 'Local Storage' operator details. The top bar shows the project 'openshift-local-storage'. The 'Details' tab is selected. The 'Provided APIs' section lists three resources: 'Local Volume', 'Local Volume Set', and 'Local Volume Discovery'. Each resource has a brief description, a 'Create instance' button, and a 'Provider' field set to 'Red Hat'. To the right, there are sections for 'Created at' (6 minutes ago), 'Links' (Documentation, GitHub URL), and 'Source Repository' (GitHub URL). At the bottom right, there is a 'Maintainers' section.

La page Operator Details répertorie les API fournies par l'opérateur et vous permet de créer des instances de ces ressources. Par exemple, à partir de la page de l'opérateur de stockage local, vous pouvez créer des instances de volume local, d'ensemble de volumes locaux et de découverte de volumes locaux.



## Références

Pour plus d'informations, reportez-vous à la documentation *Web console* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/web\\_console/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/web_console/index)

Pour plus d'informations, reportez-vous à la documentation *Operators* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/operators/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/operators/index)

## ► Exercice guidé

# Gestion des charges de travail et des opérateurs

Dans cet exercice, vous allez gérer les charges de travail du cluster avec la console Web.

## Résultats

Vous devez pouvoir utiliser la console Web OpenShift pour :

- Installer un opérateur depuis OperatorHub.
- Utiliser des fonctionnalités personnalisées fournies par un opérateur.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée les ressources nécessaires à cette activité.

```
[student@workstation ~]$ lab console-workloads start
```

## Instructions

- 1. En tant qu'utilisateur `admin`, localisez la console Web OpenShift et accédez-y.

1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

1.2. Identifiez l'URL de la console Web.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

1.3. Ouvrez un navigateur Web et accédez à l'URL de la console Web :

- <https://console-openshift-console.apps.ocp4.example.com>

1.4. Cliquez sur `localusers` et connectez-vous en tant qu'utilisateur `admin` avec le mot de passe `redhat`.

- 2. Inspectez les déploiements `openshift-console-operator` et `openshift-console`, les ensembles de répliques et les pods.

- 2.1. Dans le volet de gauche, cliquez sur **Workloads > Deployments** et sélectionnez **all projects** dans la liste de projets en haut de la page. Saisissez **console** dans le champ **Search by name**. Si l'est pas activé, cliquez sur **Show default projects** pour afficher tous les projets du cluster.

Notez qu'OpenShift dispose d'un déploiement nommé **console-operator** doté d'un seul pod dans l'espace de noms **openshift-console-operator**, qui utilise un déploiement nommé **console** dans l'espace de noms **openshift-console**.

The screenshot shows the 'Deployments' list in the OpenShift web interface. The search bar at the top has 'Name' selected and contains the value 'console'. There is a 'Create Deployment' button in the top right. Below the search bar, there are filter buttons for 'Name' and 'Namespace' (set to 'console'), and a 'Clear all filters' link. The main table has columns: Name, Namespace, Status, Labels, and Pod Selector. The first row for 'console' shows '2 of 2 pods' in status, with labels 'app=console' and 'component=ui', and a pod selector 'app=console, component=ui'. The second row for 'console-operator' shows '1 of 1 pods' in status, with no labels, and a pod selector 'name=console-operator'.

- 2.2. Dans le volet de gauche, cliquez sur **Workloads > Replica Sets** et saisissez **console** dans le champ **Search by name**.

Les déploiements déclarent un **ReplicaSet** pour s'assurer qu'un nombre donné de pods est toujours en cours d'exécution.

- 2.3. Dans la colonne Status, cliquez sur **2 of 2 pods** pour afficher la liste de pods **ReplicaSet** de la console.

### ▶ 3. Installez l'opérateur de sécurité de conteneur (CSO) à partir d'OperatorHub.

- 3.1. Dans le volet de gauche, cliquez sur **Operators > OperatorHub**, puis sur **Security** pour afficher la liste des opérateurs de sécurité disponibles depuis OperatorHub.

- 3.2. Faites défiler la page vers le bas et cliquez sur **Quay Container Security** pour afficher la page de l'opérateur, puis cliquez sur **Install**.

The screenshot shows the Red Hat OpenShift Container Platform OperatorHub interface. On the left, the navigation sidebar includes 'Administrator', 'Home', 'Operators' (selected), 'OperatorHub' (highlighted), 'Installed Operators', 'Workloads', 'Pods', 'Deployments', 'DeploymentConfigs', and 'StatefulSets'. The main content area shows a card for the 'Quay Container Security' operator. It includes the operator logo, version '3.6.4 provided by Red Hat', an 'Install' button, and detailed information about the latest version. The 'Capability level' section lists 'Basic Install', 'Seamless Upgrades', 'Full Lifecycle', 'Deep Insights' (disabled), and 'Auto Pilot' (disabled). The 'Source' section indicates it's from the 'do280 Operator Catalog'. A note explains the operator's function: 'The Quay Container Security Operator (CSO) brings Quay and Clair metadata to Kubernetes / OpenShift. Starting with vulnerability information the scope will get expanded over time. If it runs on OpenShift, the corresponding vulnerability information is shown inside the OCP Console. The Quay Container Security Operator enables cluster administrators to monitor known container image vulnerabilities in pods running on their Kubernetes cluster. The controller sets up a watch on pods in the specified namespace(s) and queries the container registry for vulnerability information. If the container registry supports image scanning, such as Quay with Clair, then the Operator will expose any vulnerabilities found via the Kubernetes API in an `ImageManifestVuln` object. This Operator requires no additional configuration after deployment, and will begin watching pods and populating `ImageManifestVulns` immediately once installed.'

- 3.3. Sélectionnez l'option **All namespaces on the cluster**, puis cliquez sur **Install** pour installer l'opérateur. Ne modifiez pas les autres champs du formulaire.

► **4.** Examinez les ressources créées par l'opérateur.

- 4.1. Dans le volet de gauche, cliquez sur **Workloads > Deployments** et inspectez la liste des déploiements. Vous remarquerez un déploiement **container-security-operator**.
- 4.2. Cliquez sur le déploiement **container-security-operator**, puis sur l'onglet **Pods** pour afficher le pod créé par ce déploiement. Cliquez sur le nom du pod pour afficher la page **Pod Details**.

► **5.** Créez un deployment pour une application Web simple.

- 5.1. Dans la liste déroulante **Project**, sélectionnez le projet **console-apps**.
- 5.2. Dans le volet de gauche, cliquez sur **Workloads > Deployments**, puis sur **Create Deployment** pour afficher l'éditeur YAML de la console Web. Mettez à jour YAML comme suit, puis cliquez sur **Create**.



**Note**

Vous pouvez copier YAML à partir du fichier `~/D0280/labs/console-workloads/deployment.yaml` sur la machine workstation.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: hello-world
  namespace: console-apps
spec:
  selector:
    matchLabels:
      app: hello-world
  replicas: 1
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
        - name: hello-world
          image: quay.io/redhattraining/hello-world-nginx:v1.0
          ports:
            - containerPort: 8080
              protocol: TCP
```

► **6.** Vérifiez le pod de déploiement à l'aide de l'opérateur de sécurité.

- 6.1. Basculez vers le projet **console-apps**.

```
[student@workstation ~]$ oc project console-apps
Now using project "console-apps" on server "https://api.ocp4.example.com:6443".
```

- 6.2. Listez les pods dans le projet console-apps.

```
[student@workstation ~]$ oc get pods
NAME                      READY   STATUS    RESTARTS   AGE
hello-world-685d5cf88-f4h9l   1/1     Running   0          16m
```

- 6.3. Vérifiez la sécurité du pod hello-world-685d5cf88-f4h9l en obtenant la ressource `imagemanifestvulns.secscan.quay.redhat.com` avec le sélecteur `console-apps/hello-world-685d5cf88-f4h9l`

```
[student@workstation ~]$ oc get imagemanifestvulns.secscan.quay.redhat.com \
>   --selector=console-apps/hello-world-685d5cf88-f4h9l \
>   -o jsonpath='{.items[*].spec.features[*].vulnerabilities[*].name}'
RHSA-2019:3352: gdb security, bug fix, and enhancement update (Low)
RHSA-2020:1635: gdb security and bug fix update (Moderate)
RHSA-2019:3352: gdb security, bug fix, and enhancement update (Low)
RHSA-2020:1635: gdb security and bug fix update (Moderate)
RHSA-2020:1864: gcc security and bug fix update (Moderate)
RHSA-2021:4386: gcc security and bug fix update (Low)
RHSA-2021:4587: gcc security update (Moderate)
...output omitted...
```

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice.

```
[student@workstation ~]$ lab console-workloads finish
```



### Important

Ne supprimez pas le projet `console-apps` ou tous travaux que vous avez effectués dans cette section. Il sera utilisé dans les sections suivantes.

L'exercice guidé est maintenant terminé.

# Examen des mesures du cluster

## Résultats

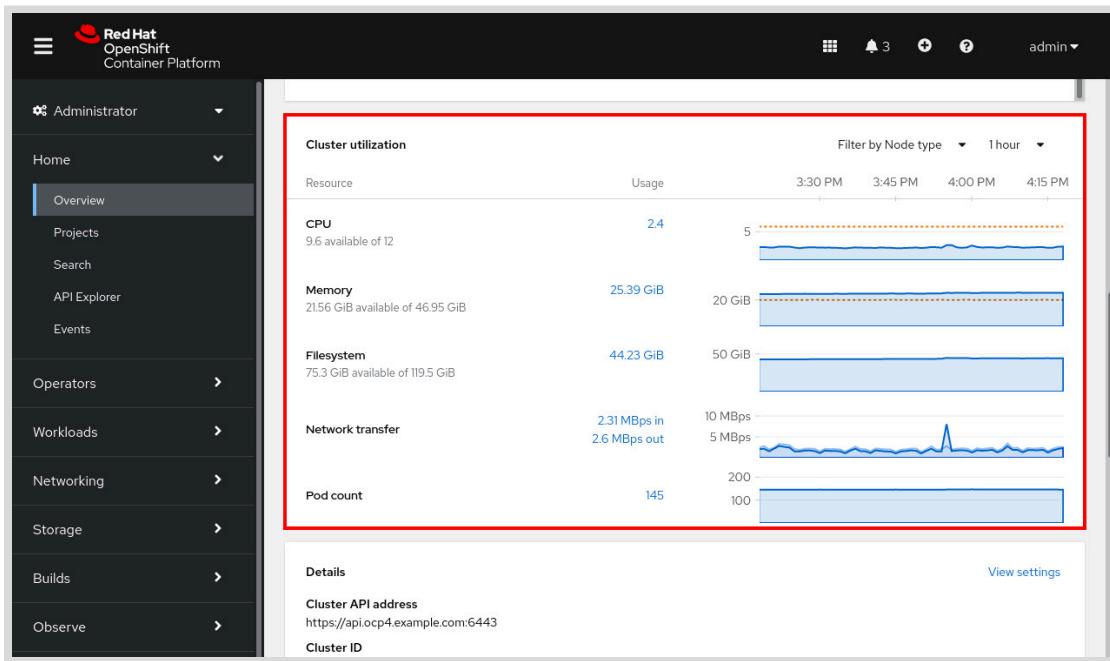
Après avoir terminé cette section, vous devez pouvoir examiner les mesures de performances et d'intégrité pour des applications et des nœuds de cluster.

## Affichage des mesures de cluster

La console Web OpenShift intègre des graphiques utiles pour visualiser les analyses des ressources et des clusters. Les administrateurs de cluster et les utilisateurs qui possèdent le rôle de cluster `view` ou `cluster-monitoring-view` peuvent accéder à la page `Home > Overview`. La page Overview affiche une série de mesures à l'échelle du cluster et fournit une vue de haut niveau de l'état de santé global du cluster.

La vue d'ensemble inclut les éléments suivants :

- Capacité du cluster actuelle en fonction de l'utilisation du processeur, de la mémoire, du stockage et du réseau.
- Graphique de série chronologique de l'utilisation totale du processeur, de la mémoire et du disque.
- La possibilité d'afficher les meilleurs consommateurs de processeur, de mémoire et de stockage.

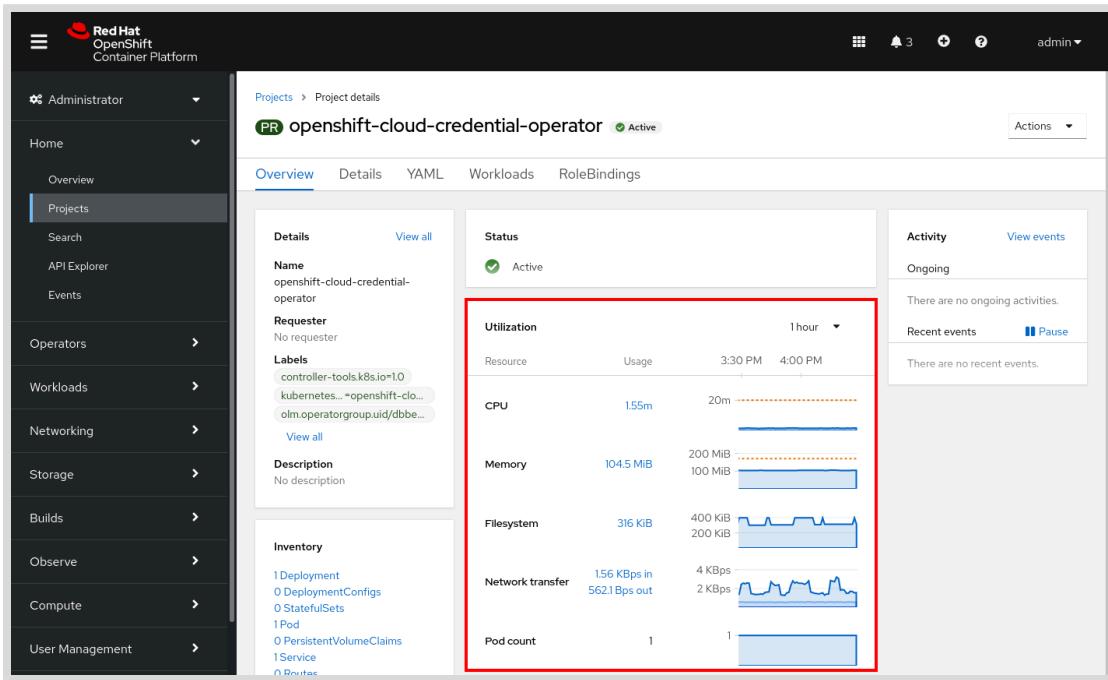


Pour chacune des ressources listées dans la section **Cluster Utilization**, les administrateurs peuvent cliquer sur le lien pour l'utilisation des ressources actuelles. Le lien affiche une fenêtre avec une décomposition des principaux consommateurs de cette ressource. Les principaux consommateurs peuvent être triés par projet, par pod ou par nœud. La liste des plus grands

consommateurs peut être utile pour identifier les pods ou nœuds problématiques. Par exemple, un pod avec une fuite de mémoire inattendue peut apparaître en haut de la liste.

## Affichage des mesures des projets

La page **Project Details** affiche les mesures qui donnent une vue d'ensemble des ressources utilisées dans le cadre d'un projet spécifique. La section **Utilization** affiche des informations sur l'utilisation des ressources telles que le processeur et la mémoire, ainsi que la possibilité d'afficher les grands consommateurs pour chaque ressource.



Toutes les mesures sont extraites de Prometheus. Cliquez sur n'importe quel graphique pour accéder à la page **Metrics**. Affichez la requête exécutée et inspectez les données en détail.

Si un quota de ressources est créé pour le projet, la demande et les limites du projet actuel s'affichent sur la page **Project Details**.

## Affichage des mesures des ressources

Lors de la résolution des problèmes, il est souvent utile d'afficher les mesures à un niveau de granularité inférieur à celui de l'ensemble du cluster ou du projet entier. La page **Pod Details** affiche les graphiques de séries chronologiques de l'utilisation du processeur, de la mémoire et du système de fichiers pour un pod donné. Une modification soudaine de ces métriques critiques, telle qu'un pic processeur causé par une charge élevée, sera visible sur cette page.

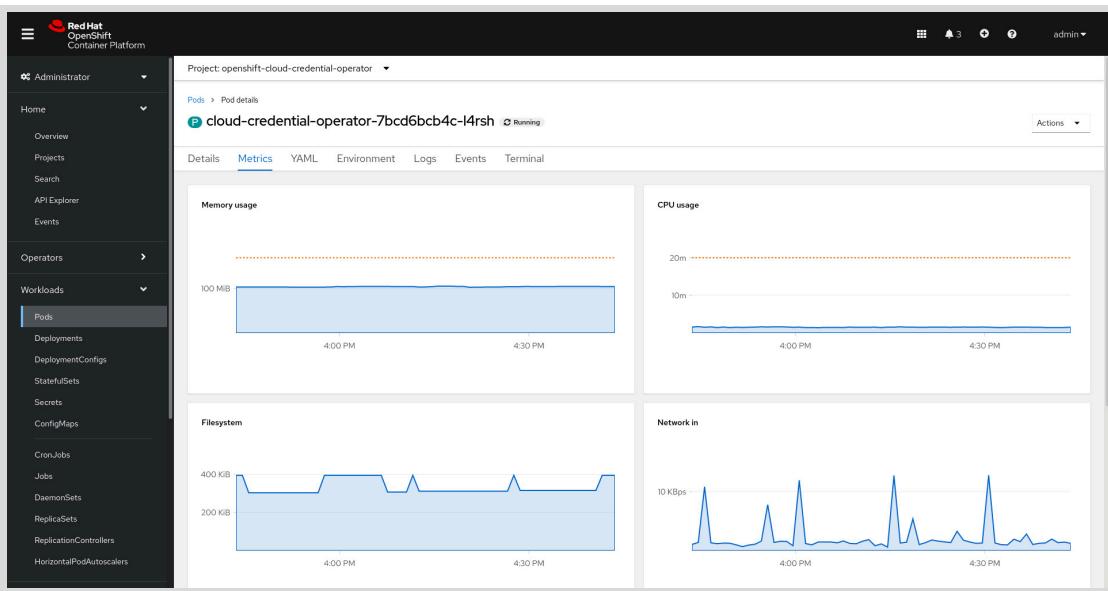


Figure 15.16: Graphiques de séries chronologiques montrant différentes mesures pour un pod

## Exécution de requêtes Prometheus dans la console Web

L'interface utilisateur Prometheus est un outil riche en fonctionnalités permettant de visualiser les mesures et de configurer les alertes. La console Web OpenShift fournit une interface permettant d'exécuter des requêtes Prometheus directement depuis celle-ci.

Pour effectuer une requête, accédez à **Observe > Metrics**, saisissez une expression Prometheus Query Language dans le champ de texte, puis cliquez sur **Run Queries**. Les résultats de la requête sont affichés sous la forme d'un graphique de série chronologique :

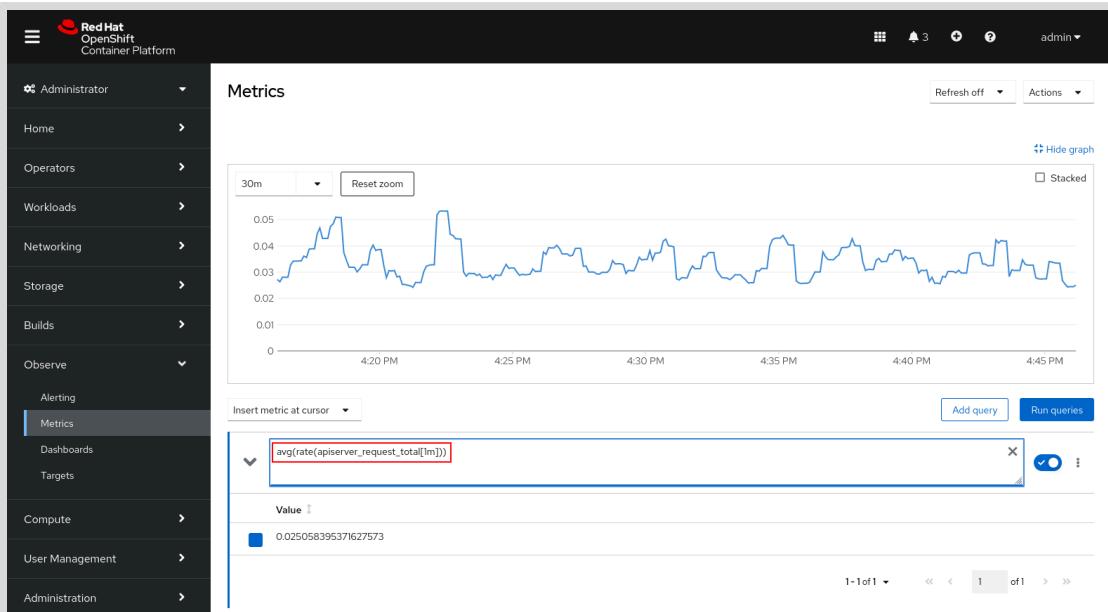


Figure 15.17: Utilisation d'une requête Prometheus pour afficher un graphique de séries chronologiques



### Note

Prometheus Query Language n'est pas abordé en détail dans ce cours. Reportez-vous aux références ci-dessous pour obtenir un lien vers la documentation officielle.



### Références

Pour plus d'informations, reportez-vous à la documentation *Monitoring* de Red Hat OpenShift Container Platform 4.10, disponible à l'adresse  
[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/monitoring/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/monitoring/index)

### Interrogation de Prometheus

<https://prometheus.io/docs/prometheus/latest/querying/basics/>

## ► Exercice guidé

# Examen des mesures du cluster

Dans cet exercice, vous allez examiner la page de mesure et le tableau de bord sur la console Web.

### Résultats

Vous serez capable d'utiliser la console Web OpenShift de Red Hat pour :

- Afficher les mesures de cluster, de projets, de pods et de nœuds.
- Identifier un pod consommant des quantités importantes de mémoire ou de processeur.

### Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée les ressources nécessaires à cet exercice.

```
[student@workstation ~]$ lab console-metrics start
```

### Instructions

- 1. En tant qu'utilisateur `admin`, localisez la console Web OpenShift et accédez-y.

- 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
>   https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Identifiez l'URL de la console Web.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Ouvrez un navigateur Web et accédez à l'URL de la console Web :

- <https://console-openshift-console.apps.ocp4.example.com>

- 1.4. Cliquez sur `localusers` et connectez-vous en tant qu'utilisateur `admin` avec le mot de passe `redhat`.

- 2. Dans cet exercice guidé, vous verrez comment les modifications de la charge sont affichées dans la console Web. Commencez par observer les métriques de référence de l'intégrité sur les pages **Overview**, **Pod Details** et **Project Details**.
- 2.1. Cliquez sur **Home > Overview** pour afficher la page **Overview**. Faites défiler vers le bas jusqu'à la section **Cluster Utilization** qui affiche le graphique historique des séries chronologiques du processeur, de la mémoire et de l'utilisation du disque du cluster.
  - 2.2. Pour chaque ressource de la table, telle que **CPU**, **Memory** ou **Filesystem**, cliquez sur le lien d'utilisation pour afficher les **Top Consumers** de cette ressource. Par défaut, la fenêtre filtre les plus grands consommateurs par projet, mais vous pouvez filtrer par pod ou par nœud à la place.
  - 2.3. Cliquez sur le lien d'utilisation de **Memory**, filtrez les plus grands consommateurs par pod, puis cliquez sur le nom du pod qui consomme le plus de ressources de mémoire.

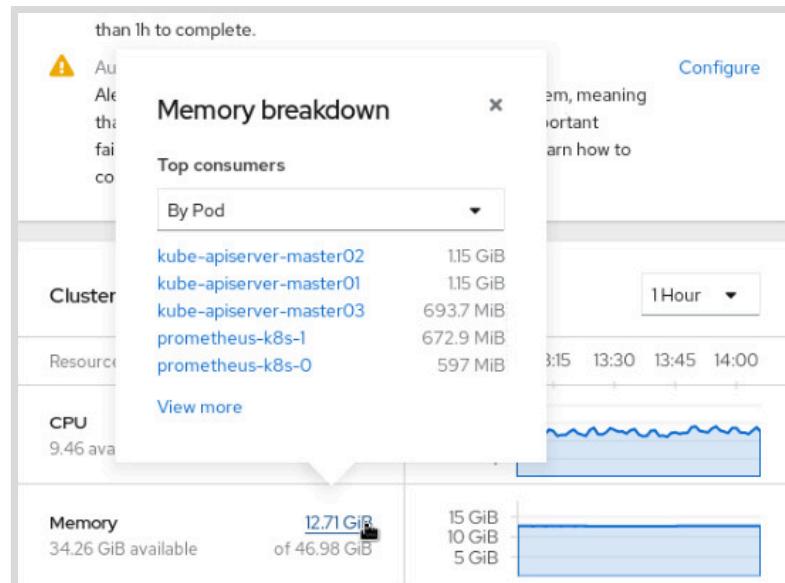


Figure 15.18: Répartition de la mémoire : plus grands consommateurs par pod

- 2.4. Cliquez sur **Metrics** pour afficher des graphiques historiques de séries chronologiques d'utilisation de la mémoire, du processeur et du système de fichiers.
  - 2.5. Cliquez sur **Home > Projects**, puis sur **console-apps** pour afficher la page **Project Details**. Remarquez la section **Utilization** qui affiche les métriques des charges de travail exécutées dans le projet **console-apps**. Les liens de la colonne **Usage** ouvrent des fenêtres affichant les pods qui consomment le plus de ressources. Les charges de travail sont exécutées de manière sécurisée dans leurs limites.
  - 2.6. Faites défiler vers le bas jusqu'à la section **Resource Quotas** qui affiche l'utilisation actuelle du processeur et de la mémoire par rapport au quota alloué.
- 3. Rechercher et passer en revue les mesures de référence de l'intégrité d'un nœud de calcul.
- 3.1. Cliquez sur **Compute > Nodes**, puis sur l'un des nœuds de la liste.
  - 3.2. Dans la section **Utilization**, remarquez les graphiques de séries chronologiques qui affichent les métriques du nœud sélectionné.

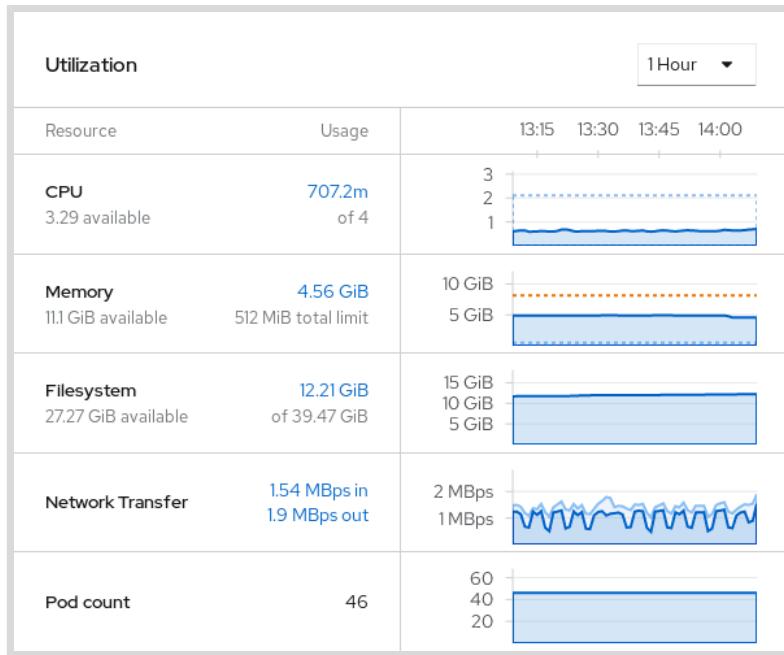


Figure 15.19: Graphiques de séries chronologiques montrant différentes mesures pour un nœud

- 4. Sur la machine `workstation`, exécutez le script `load.sh` pour générer une charge sur le déploiement `books` d'exemple. L'application contient intentionnellement une fuite de mémoire qui utilise plusieurs mégaoctets de RAM avec chaque requête sur son chemin d'accès `/leak`.

- 4.1. Dans un terminal sur la machine `workstation`, exécutez la commande suivante.

```
[student@workstation ~]$ ~/DO280/labs/console-metrics/load.sh
```

- 5. Dans la console Web OpenShift, observez le changement dans les mesures et identifiez le pod problématique. Les données affichées dans la console Web sont automatiquement actualisées, il n'est donc pas nécessaire de recharger la page.

- 5.1. Cliquez sur **Home > Projects**, puis sur `console-apps` pour afficher la page **Project Details**. Observez le graphique de séries chronologiques **Memory Usage** pour surveiller les modifications.

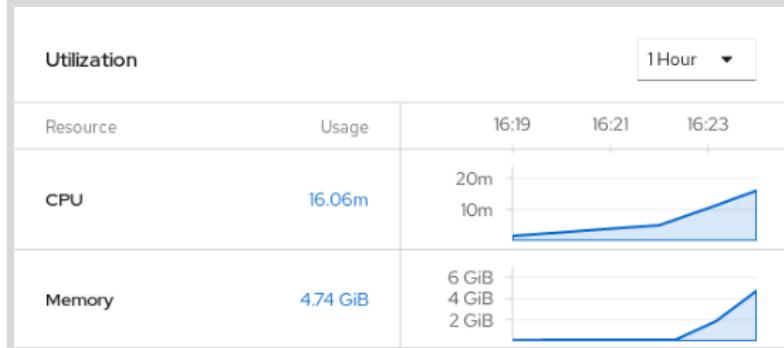


Figure 15.20: Graphiques d'utilisation indiquant une éventuelle fuite de mémoire

La perte de mémoire peut prendre une ou deux minutes avant d'être suffisamment importante pour être visible. Bien que le processeur et la mémoire augmentent, l'utilisation totale du processeur reste faible.

- 5.2. Cliquez sur **Home > Overview** pour afficher la page **Overview**. La mémoire consommée par le test de charge peut être trop petite pour être remarquée sur un cluster de taille importante, mais la fenêtre **Memory breakdown** (triée par pod) fournit une liste pratique des pods utilisant le plus de mémoire. Affichez la fenêtre **Memory breakdown** en cliquant sur le lien d'utilisation de **Memory**. Triez les plus grands consommateurs par pod.

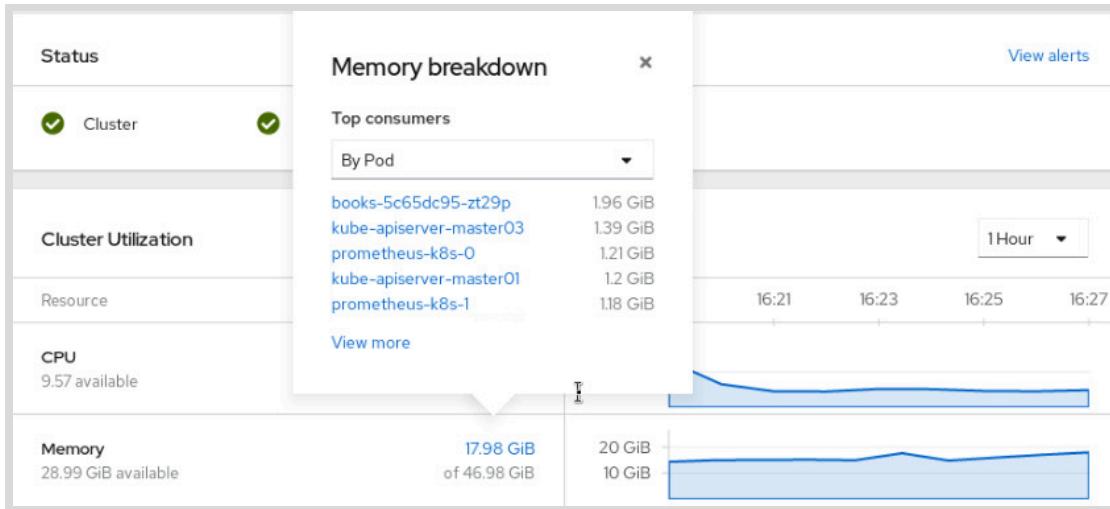
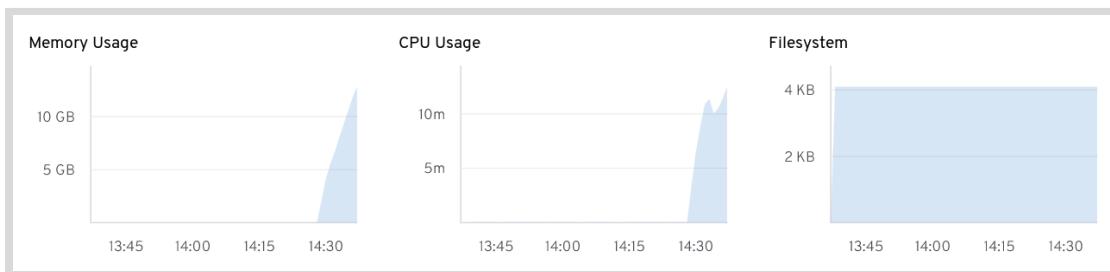


Figure 15.21: Le pod books est un grand consommateur de mémoire.

Le pod books apparaît en haut ou dans les premiers de la liste. S'il n'est pas dans la liste, vous devrez peut-être attendre une minute de plus que le script de charge soit terminé.

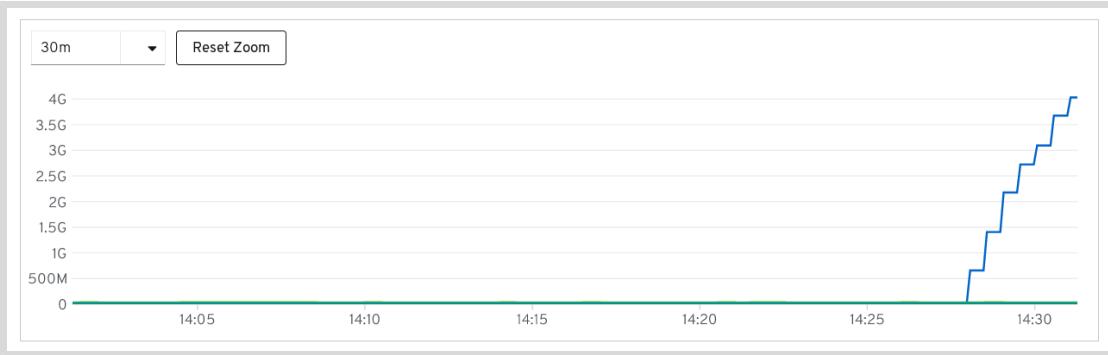
- 5.3. Cliquez sur le lien de pod books dans la fenêtre **Memory breakdown** pour accéder à la page **Pod Details**. Cliquez sur le lien **Metrics** pour afficher les mesures de pod, notez que la fuite de mémoire progressive est visible dans le graphique de séries chronologiques **Memory Usage**.



- 5.4. Cliquez sur **Observe > Metrics** pour afficher la page **Metrics** de la console Web. Saisissez la requête Prometheus suivante dans le champ d'entrée d'expression :

```
avg(container_memory_working_set_bytes{namespace='console-apps'}) BY (pod)
```

Cliquez sur **Run Queries** pour afficher les résultats dans la console Web OpenShift.



► 6. Supprimez le projet `console-apps` et arrêtez le test de charge.

- 6.1. Cliquez sur **Home > Projects**, puis sur **Delete Project** dans le menu situé à l'extrême droite de la ligne `console-apps`.

Project	Status	Owner	Labels	Actions
PR console-apps	Active	admin	No labels	⋮
PR default	Active	No requester	No labels	Edit Project
PR kube-node-lease	Active	No requester	No labels	⋮
PR kube-public	Active	No requester	No labels	⋮

- 6.2. Dans la boîte de dialogue **Delete Project**, saisissez `console-apps`, puis cliquez sur **Delete**.
- 6.3. Si le script `load.sh` est toujours en cours d'exécution sur le terminal `Workstation`, appuyez sur `Ctrl+C` dans le terminal pour arrêter le test de charge.

## Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab console-metrics finish
```

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Gestion d'un cluster avec la console Web

Au cours de cet atelier, vous allez gérer le cluster OpenShift à l'aide de la console Web.

### Résultats

Vous devez pouvoir utiliser la console Web OpenShift pour :

- Modifier un secret afin d'ajouter des entrées htpasswd pour les nouveaux utilisateurs.
- Configurer un nouveau projet avec des contrôles d'accès basés sur les rôles et des quotas de ressources.
- Utiliser un opérateur OperatorHub pour déployer une base de données.
- Créer un déploiement, un service et une route pour une application Web.
- Résoudre les problèmes liés à une application en utilisant les journaux et événements.

### Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée un répertoire pour les fichiers de l'exercice.

```
[student@workstation ~]$ lab console-review start
```

### Instructions

1. Connectez-vous à la console Web OpenShift en tant qu'utilisateur `admin`.
2. Ajoutez des entrées `htpasswd` au secret `localusers` pour les utilisateurs nommés `dba` et `tester` à l'aide du mot de passe `redhat`.
3. Créez un nouveau groupe `app-team` contenant les utilisateurs `developer` et `dba`.
4. Créez un projet `console-review` avec une liaison de rôle `view` pour l'utilisateur `tester` et une liaison de rôle `edit` pour le groupe `app-team`. Définissez un quota de ressources limitant le projet à deux pods.
5. En tant qu'utilisateur `dba`, déployez une instance de MySQL Database dans le projet `console-review`. Définissez `database` comme nom d'application et `famous` comme nom d'utilisateur, mot de passe et nom de base de données. Utilisez l'image `registry.redhat.io/rhel8/mysql-80:1`.
6. En tant qu'utilisateur `developer`, créez un déploiement, un service et une route dans le projet `console-review` avec les problèmes que vous allez résoudre à la prochaine étape. Utilisez l'image `quay.io/redhat-training/famous-quotes:2.1`, un réplica et nommez toutes les nouvelles ressources `famous-quotes`. Lorsqu'elle est correctement configurée, l'application `famous-quotes` se connecte à la base de données MySQL et affiche une liste des citations célèbres.

**Note**

Vous pouvez copier les ressources YAML de déploiement et de service à partir du répertoire `~/D0280/labs/console-review` sur la machine `workstation`.

Spécifiez les variables d'environnement suivantes dans le déploiement :

**Variables d'environnement de déploiement**

Name	Valeur
QUOTES_HOSTNAME	database
QUOTES_USER	famous
QUOTES_DATABASE	famous
QUOTES_PASSWORD	famous

**Important**

Vous allez résoudre les problèmes liés au déploiement à l'étape suivante.

7. Rechercher et résoudre les problèmes de déploiement.
8. Accédez au site Web `famous-quotes` dans un navigateur et observez l'application en cours d'utilisation.

## Évaluation

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab console-review grade
```

## Fin

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour effectuer cet exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab console-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Gestion d'un cluster avec la console Web

Au cours de cet atelier, vous allez gérer le cluster OpenShift à l'aide de la console Web.

## Résultats

Vous devez pouvoir utiliser la console Web OpenShift pour :

- Modifier un secret afin d'ajouter des entrées htpasswd pour les nouveaux utilisateurs.
- Configurer un nouveau projet avec des contrôles d'accès basés sur les rôles et des quotas de ressources.
- Utiliser un opérateur OperatorHub pour déployer une base de données.
- Créer un déploiement, un service et une route pour une application Web.
- Résoudre les problèmes liés à une application en utilisant les journaux et événements.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

Cette commande garantit que l'API du cluster est accessible et crée un répertoire pour les fichiers de l'exercice.

```
[student@workstation ~]$ lab console-review start
```

## Instructions

1. Connectez-vous à la console Web OpenShift en tant qu'utilisateur `admin`.

- 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p redhat \
> https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Identifiez l'URL de la console Web.

```
[student@workstation ~]$ oc whoami --show-console
https://console-openshift-console.apps.ocp4.example.com
```

- 1.3. Ouvrez un navigateur Web et accédez à l'URL de la console Web :

- <https://console-openshift-console.apps.ocp4.example.com>

- 1.4. Cliquez sur **localusers** et connectez-vous en tant qu'utilisateur **admin** avec le mot de passe **redhat**.
2. Ajoutez des entrées **htpasswd** au secret **localusers** pour les utilisateurs nommés **dba** et **tester** à l'aide du mot de passe **redhat**.
  - 2.1. Dans l'interface utilisateur Web de Red Hat OpenShift Container Platform, cliquez sur **Workloads > Secrets**, puis sélectionnez **openshift-config** dans la liste de recherche **Project** pour afficher les secrets du projet **openshift-config**. Si les projets par défaut sont masqués, cliquez sur **Show default projects** pour afficher tous les projets du cluster.
  - 2.2. Faites défiler vers le bas de la page et cliquez sur le lien **localusers** pour afficher les **localusers** de **Secret Details**.
  - 2.3. Cliquez sur **Actions > Edit Secret** en haut de la page pour naviguer jusqu'à l'outil **Edit Key/Value Secret**.
  - 2.4. Utilisez un terminal sur la machine **workstation** pour générer une entrée **htpasswd** chiffrée pour les deux utilisateurs.

```
[student@workstation ~]$ htpasswd -n -b dba redhat
dba:$apr1$YF4ack.9$qho0THlWTC.cLBByNEHDaV

[student@workstation ~]$ htpasswd -n -b tester redhat
tester:$apr1$XdTSqET7$i0hkC5bIs7PhYUm2KhiI.0
```

- 2.5. Ajoutez la sortie de terminal depuis les commandes **htpasswd** à la valeur **htpasswd** dans l'éditeur de secrets de la console Web OpenShift, puis cliquez sur **Save**.

```
admin:$apr1$Au9.fFr$0k5wvUBd3eeBt0baa77.dae
leader:$apr1$/abo4Hybn7a.tG5ZoOBn.QwefXckiy1
developer:$apr1$RjqTY4cv$xql3.BQfg42moSxwnTNkh.
dba:$apr1$YF4ack.9$qho0THlWTC.cLBByNEHDaV
tester:$apr1$XdTSqET7$i0hkC5bIs7PhYUm2KhiI.0
```

3. Créez un nouveau groupe **app-team** contenant les utilisateurs **developer** et **dba**.
  - 3.1. Cliquez sur **User Management > Groups**, puis sur **Create Group**. Utilisez l'éditeur YAML pour définir une ressource Group comme suit, puis cliquez sur **Create** pour ajouter le nouveau groupe **app-team**. Mettez à jour YAML comme suit, puis cliquez sur **Create**.

```
apiVersion: user.openshift.io/v1
kind: Group
metadata:
  name: app-team
users:
  - developer
  - dba
```

4. Créez un projet **console-review** avec une liaison de rôle **view** pour l'utilisateur **tester** et une liaison de rôle **edit** pour le groupe **app-team**. Définissez un quota de ressources limitant le projet à deux pods.

- 4.1. Cliquez sur **Home > Projects** pour afficher la page Projects, puis cliquez sur **Create Project**. Saisissez **console-review** dans le champ **Name**, puis indiquez un **Display Name** et une **Description** facultatifs. Cliquez sur **Create**.
- 4.2. Cliquez sur **User Management > RoleBindings**, puis sur **Create binding**. Remplissez le formulaire comme suit afin de créer le Role Binding d'espace de noms pour le groupe **app-team**. Cliquez sur **Create** pour créer le RoleBinding d'espace de noms.

#### Formulaire App Team Role Binding

Champ	Valeur
Binding Type	Namespace Role Binding (RoleBinding)
Name	app-team
Namespace	console-review
Role Name	edit
Subject	Group
Subject Name	app-team

- 4.3. Cliquez sur le lien **RoleBindings** pour revenir à la page **RoleBindings**, puis cliquez sur **Create Binding**. Remplissez le formulaire comme suit afin de créer le Role Binding d'espace de noms pour l'utilisateur **tester**. Cliquez sur **Create** pour créer le RoleBinding d'espace de noms.

#### Formulaire Tester Role Binding

Champ	Valeur
Binding Type	Namespace Role Binding (RoleBinding)
Name	tester
Namespace	console-review
Role Name	view
Subject	User
Subject Name	tester

- 4.4. Cliquez sur **Administration > Resource Quotas**, puis sur **Create Resource Quota**. Modifiez le document YAML pour spécifier une limite de quatre pods comme suit. Supprimez les demandes et limites de processeur et de mémoire, puis cliquez sur **Create**.

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: quota
  namespace: console-review
spec:
  hard:
    pods: '2'
```

5. En tant qu'utilisateur dba, déployez une instance de MySQL Database dans le projet **console-review**. Définissez **database** comme nom d'application et **famous** comme nom d'utilisateur, mot de passe et nom de base de données. Utilisez l'image **registry.redhat.io/rhel8/mysql-80:1**.

- 5.1. Connectez-vous en tant qu'utilisateur dba avec le mot de passe **redhat**.

```
[student@workstation ~]$ oc login -u dba -p redhat \
> https://api.ocp4.example.com:6443
...output omitted...
```

- 5.2. Basculez vers le projet **console-review**.

```
[student@workstation ~]$ oc project console-review
Now using project "console-review" on server "https://api.ocp4.example.com:6443".
```

- 5.3. Créez un déploiement de base de données à l'aide de l'image de conteneur située à l'emplacement suivant : **registry.redhat.io/rhel8/mysql-80:1**.

```
[student@workstation ~]$ oc new-app --name database \
> --image registry.redhat.io/rhel8/mysql-80:1      \
> -e MYSQL_USER=famous      \
> -e MYSQL_PASSWORD=famous  \
> -e MYSQL_DATABASE=famous
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "database" created
  deployment.apps "database" created
  service "database" created
--> Success
...output omitted...
```

6. En tant qu'utilisateur **developer**, créez un déploiement, un service et une route dans le projet **console-review** avec les problèmes que vous allez résoudre à la prochaine étape. Utilisez l'image **quay.io/redhattraining/famous-quotes:2.1**, un réplica et nommez toutes les nouvelles ressources **famous-quotes**. Lorsqu'elle est correctement configurée, l'application **famous-quotes** se connecte à la base de données MySQL et affiche une liste des citations célèbres.

**Note**

Vous pouvez copier les ressources YAML de déploiement et de service à partir du répertoire `~/DO280/labs/console-review` sur la machine `workstation`.

Spécifiez les variables d'environnement suivantes dans le déploiement :

**Variables d'environnement de déploiement**

Name	Valeur
QUOTES_HOSTNAME	database
QUOTES_USER	famous
QUOTES_DATABASE	famous
QUOTES_PASSWORD	famous

**Important**

Vous allez résoudre les problèmes liés au déploiement à l'étape suivante.

- 6.1. Cliquez sur **admin > Log out**, puis connectez-vous en tant qu'utilisateur **developer** avec le mot de passe **developer**.
- 6.2. Passez à la perspective **Administrator**. Cliquez ensuite sur le projet **console-review** pour basculer vers le projet **console-review**.
- 6.3. Cliquez sur **Workloads > Deployments**, puis sur **Create Deployment** pour afficher l'éditeur YAML de la console Web. Mettez à jour YAML comme suit, puis cliquez sur **Create**.

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: famous-quotes
  namespace: console-review
spec:
  selector:
    matchLabels:
      app: famous-quotes
  replicas: 2
  template:
    metadata:
      labels:
        app: famous-quotes
    spec:
      containers:
        - name: famous-quotes
          image: 'quay.io/redhattraining/famous-quotes:2.1'
          ports:
```

```
- containerPort: 8000
  protocol: TCP
env:
- name: QUOTES_HOSTNAME
  value: database
- name: QUOTES_USER
  value: famous
- name: QUOTES_DATABASE
  value: famous
- name: QUOTES_PASSWORD
  value: famous
```

**Note**

Vous pouvez utiliser le manifeste YAML à partir de ~/D0280/labs/console-review/deployment.yaml pour créer la ressource.

- 6.4. Cliquez sur **Networking > Services**, puis sur **Create Service** pour afficher l'éditeur YAML de la console Web. Mettez à jour YAML comme suit, puis cliquez sur **Create**.

```
kind: Service
apiVersion: v1
metadata:
  name: famous-quotes
  namespace: console-review
spec:
  selector:
    app: famous-quotes
  ports:
    - protocol: TCP
      port: 8000
      targetPort: 8000
```

**Note**

Vous pouvez utiliser le manifeste YAML à partir de ~/D0280/labs/console-review/service.yaml pour créer la ressource.

- 6.5. Cliquez sur **Networking > Routes**, puis sur **Create Route**. Complétez le formulaire comme suit, en ne modifiant pas les autres champs, puis cliquez sur **Create**.

**Formulaire Create Route**

Champ	Valeur
Name	famous
Service	famous-quotes
Target Port	8000 → 8000 (TCP)

7. Rechercher et résoudre les problèmes de déploiement.

- 7.1. Cliquez sur **developer** > **Log out**, puis connectez-vous en tant qu'utilisateur **admin** avec le mot de passe **redhat**.
- 7.2. Cliquez sur **Home** > **Events**, puis sélectionnez **console-review** dans le filtre de la liste de projets en haut de la page. Notez l'erreur de quota famous-quotes.

```
(combined from similar events): Error creating: pods "famous-quotes-8449f7dc6d-k94xq" is forbidden: exceeded quota: quota, requested: pods=1, used: pods=2, limited: pods=2
```

- 7.3. Cliquez sur **Administration** > **Resource Quotas**, puis sélectionnez **console-review** dans la liste de filtres **Project**.
- 7.4. Cliquez sur le lien **quota** dans la liste des quotas de ressources, puis cliquez sur l'onglet **YAML**. Modifiez la **spec** pour spécifier une limite de quatre pods comme suit, puis cliquez sur **Save**.

```
kind: ResourceQuota
apiVersion: v1
metadata:
  name: quota
  namespace: console-review
  ...output omitted...
spec:
  hard:
    pods: '4'
  ...output omitted...
```



### Note

Le projet requiert un pod pour la réplique spécifiée de famous-quotes et un pod supplémentaire afin de déployer une modification.

- 7.5. Cliquez sur **Workloads** > **Pods** et passez en revue la liste des pods. Veuillez compter entre une et deux minutes pour que le pod **famous-quotes** apparaisse dans la liste.
8. Accédez au site Web famous-quotes dans un navigateur et observez l'application en cours d'utilisation.
  - 8.1. Cliquez sur **Networking** > **Routes**, puis sur le nom de route **famous** et enfin sur le lien dans la colonne **Location**. Firefox ouvre un nouvel onglet qui affiche une liste des citations célèbres (famous quotes).

## Évaluation

En tant qu'utilisateur **student** sur la machine **workstation**, utilisez la commande **lab** pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab console-review grade
```

## Fin

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour effectuer cet exercice. Il s'agit d'un point important pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

```
[student@workstation ~]$ lab console-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- La console Web OpenShift fournit une interface utilisateur graphique permettant la visualisation et la gestion des ressources OpenShift.
- Certaines ressources sont dotées d'une page spécialisée qui rend la création et la modification de ressources plus pratique que l'écriture manuelle de YAML, comme l'éditeur **Edit Key/Value Secret** qui gère automatiquement le codage et le décodage Base64.
- Vous pouvez installer les opérateurs de partenaire et de communauté depuis la page **OperatorHub** intégrée.
- Les mesures à l'échelle du cluster, telles que l'utilisation du processeur, de la mémoire et du stockage, sont affichées sur la page **Dashboards**.
- Les pages **Project Details** affichent les mesures spécifiques au projet, telles que les dix principaux consommateurs de mémoire par pod et l'utilisation actuelle des quotas de ressources.



# Révision complète

## Objectif

Tâches de révision depuis *Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II*

## Résultats

- Tâches de révision depuis *Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II*

## Sections

- Révision complète

## Ateliers

- Résoudre les problèmes relatifs aux applications et à un cluster OpenShift
- Configurer un modèle de projet avec des restrictions de ressources et de réseau

# Révision complète

---

## Résultats

Après avoir terminé cette section, vous aurez révisé et rafraîchi les connaissances et les compétences acquises dans le cadre du cours *Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II*.

## Révision de Administration de conteneurs, de Kubernetes et de Red Hat OpenShift II

Avant de commencer la révision complète de ce cours, vous devez être familiarisé avec les rubriques abordées dans chaque chapitre.

Vous pouvez vous référer aux précédentes sections du manuel pour en savoir plus.

### **Chapitre 1, Présentation de la technologie de conteneur**

Décrire la façon dont les logiciels peuvent s'exécuter dans des conteneurs orchestrés par Red Hat OpenShift Container Platform.

- Décrire la différence entre les applications conteneur et les déploiements traditionnels.
- Décrire les principes de base de l'architecture des conteneurs.
- Décrire les avantages de l'orchestration d'applications et d'OpenShift Container Platform.

### **Chapitre 2, Création de services en conteneur**

Déployer un serveur en utilisant la technologie des conteneurs.

- Crée un serveur de base de données à partir d'une image de conteneur.

### **Chapitre 3, Gestion des conteneurs**

Utiliser des images de conteneur préétablies pour créer et gérer des services en conteneur.

- Gérer le cycle de vie d'un conteneur, de la création à la suppression.
- Enregistrer les données d'application de conteneur avec le stockage persistant.
- Décrire la façon d'utiliser la redirection de port pour accéder à un conteneur.

### **Chapitre 4, Gestion des images de conteneur**

Gérer le cycle de vie d'une image de conteneur, de la création à la suppression.

- Rechercher et extraire des images à partir de registres distants.
- Exporter, importer et gérer les images de conteneur localement et dans un registre.

## **Chapitre 5, Cr éation d'images de conteneur personnalisées**

Concevoir et coder un Containerfile pour cr éer une image de conteneur personnalisée.

- Décrire les approches de cr éation des images de conteneur personnalisées.
- Cr éer une image de conteneur à l'aide des commandes Containerfile courantes.

## **Chapitre 6, D éploiement d'applications en conteneur dans OpenShift**

Déployer des applications conteneur uniques sur la plateforme OpenShift Container Platform.

- Décrire l'architecture de Kubernetes et de Red Hat OpenShift Container Platform.
- Cr éer des ressources Kubernetes standard.
- Cr éer une route vers un service.

## **Chapitre 7, D éploiement d'applications multiconteneurs**

Déployer des applications en conteneur à l'aide de plusieurs images de conteneur.

- Décrire les considérations relatives à la conteneurisation des applications avec plusieurs images de conteneur.

## ► Open Lab

# Résoudre les problèmes relatifs aux applications et à un cluster OpenShift

Dans le cadre de cette révision, vous allez permettre aux développeurs d'accéder à un cluster et de résoudre les problèmes liés aux déploiements d'applications.

### Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Crée un projet.
- Effectuer un test de fumée du cluster OpenShift en créant une application à l'aide du processus source-to-image.
- Créer des applications à l'aide de la ressource de déploiement.
- Utiliser le fournisseur d'identité HTPasswd pour gérer les utilisateurs.
- Créer et gérer des groupes.
- Gérer le RBAC et la SCC des groupes et des utilisateurs.
- Gérer les secrets pour les bases de données et les applications.
- Résoudre les problèmes courants.

### Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande :

- Garantit que l'API du cluster est accessible.
- Supprime les utilisateurs et les groupes existants.
- Supprime les fournisseurs d'identité existants.
- Supprime la liaison de rôle de cluster `cluster-admin` de l'utilisateur `admin`.
- Garantit que les utilisateurs authentifiés peuvent créer des projets.

```
[student@workstation ~]$ lab review-troubleshoot start
```

### Instructions

Réalisez les tâches suivantes :

1. En tant qu'utilisateur `kubeadmin`, créez le projet `review-troubleshoot`. Le mot de passe de l'utilisateur `kubeadmin` se trouve dans le fichier `/usr/local/etc/ocp4.config` sur la ligne `RHT_OCP4_KUBEADM_PASSWD`. Effectuez toutes les tâches suivantes dans le projet `review-troubleshoot`.
2. Effectuez un test de fumée du cluster pour vérifier la fonctionnalité de base du cluster. Utilisez un `deployment` pour créer une application nommée `hello-world-nginx`. Le code source de l'application se trouve dans le sous-répertoire `hello-world-nginx` du référentiel.
  - <https://github.com/RedHatTraining/DO280-apps>

Créez une route pour l'application, utilisez `hello-world.apps.ocp4.example.com` comme nom d'hôte. Vérifiez que l'application répond aux requêtes externes.

3. Configurez le cluster afin d'utiliser un fournisseur d'identité HTPasswd. Le nom du fournisseur d'identités est `cluster-users`. Le fournisseur d'identités lit les informations d'identification `htpasswd` stockées dans le secret `comprevue-users`. Assurez-vous que quatre comptes d'utilisateurs existent : `admin`, `leader`, `developer` et `qa-engineer`. Tous les comptes d'utilisateurs doivent utiliser `review` comme mot de passe. Ajoutez le rôle `cluster-admin` à l'utilisateur `admin`.
4. En tant qu'utilisateur `admin`, créez trois groupes d'utilisateurs : `leaders`, `developers` et `qa`. Affectez l'utilisateur `leader` au groupe `leaders`, l'utilisateur `developer` aux groupes `developers` et l'utilisateur `qa-engineer` au groupe `qa`. Affectez des rôles à chaque groupe :
  - Affectez le rôle `self-provisioner` au groupe `leaders`, ce qui permet aux membres de créer des projets. Pour que ce rôle soit efficace, vous devez également supprimer la capacité de tout utilisateur authentifié à créer de nouveaux projets.
  - Affectez le rôle `edit` au groupe `developers` pour le projet `review-troubleshoot` uniquement, ce qui permet aux membres de créer et de supprimer des ressources de projet.
  - Affectez le rôle `view` au groupe `qa` pour le projet `review-troubleshoot` uniquement, ce qui offre aux membres un accès en lecture aux ressources de projet.
5. En tant qu'utilisateur `developer`, utilisez un déploiement pour créer une application nommée `mysql` dans le projet `review-troubleshoot`. Utilisez l'image disponible dans `registry.redhat.io/rhel8/mysql-80:1-139`. Cette application fournit un service de base de données partagé pour d'autres applications de projet. Créez un secret générique appelé `mysql` avec `password` comme clé et `r3dh4t123` comme valeur. Définissez les variables d'environnement `MYSQL_ROOT_` à partir des valeurs du secret `mysql`. Configurez l'application de base de données `mysql` pour monter une revendication de volume persistant (PVC) dans le répertoire `/var/lib/mysql/data` à l'intérieur du pod. La PVC doit avoir une taille de 2 GB et ne doit demander que le mode d'accès `ReadWriteOnce`.

6. En tant qu'utilisateur `developer`, utilisez un `deployment` pour créer une application nommée `wordpress`. Créez l'application dans le projet `review-troubleshoot`. Utilisez l'image disponible dans `quay.io/redhattraining/wordpress:5.7-php7.4-apache`.

L'application Wordpress nécessite la définition de plusieurs variables d'environnement lors de l'exécution de la commande `oc new-app` :

- `WORDPRESS_DB_HOST` avec la valeur `mysql`.
- `WORDPRESS_DB_NAME` pour avoir la valeur `wordpress`.
- `WORDPRESS_DB_USER` pour avoir la valeur `root`.
- `WORDPRESS_USER` avec la valeur `wpuser`.
- `WORDPRESS_PASSWORD` avec la valeur `wppass`.
- `WORDPRESS_TITLE` pour avoir la valeur `review-troubleshoot`.
- `WORDPRESS_URL` avec la valeur `wordpress.apps.ocp4.example.com`.
- `WORDPRESS_EMAIL` avec la valeur `student@redhat.com`.

Définissez les variables d'environnement `WORDPRESS_DB_*` pour récupérer le mot de passe de la base de données à partir du secret `mysql`.

L'application `wordpress` requiert la contrainte de contexte de sécurité `anyuid`. Créez un compte de service nommé `wordpress-sa`, puis attribuez-lui la contrainte de contexte de sécurité `anyuid`. Configurez le déploiement `wordpress` pour utiliser le compte de service `wordpress-sa`.

L'application `wordpress` requiert également l'existence de la base de données `WORDPRESS_DB_NAME` sur le serveur de base de données. Créez une base de données vide nommée `wordpress` en exécutant la commande SQL `CREATE DATABASE wordpress` dans le pod MySQL.

Créez une route pour l'application, utilisez `wordpress.apps.ocp4.example.com` comme nom d'hôte. Si vous déployez correctement l'application, un assistant d'installation s'affiche lorsque vous accédez à l'application depuis un navigateur.

7. En tant qu'utilisateur `developer`, déployez l'application `famous-quotes` dans le projet `review-troubleshoot` à l'aide du script `~/D0280/labs/review-troubleshoot/deploy_famous-quotes.sh`. Ce script crée la base de données `defaultdb` et les ressources définies dans le fichier `~/D0280/labs/review-troubleshoot/famous-quotes.yaml`.

Utilisez le secret `mysql` afin d'initialiser les variables d'environnement pour le déploiement `famous-quotes` avec le préfixe `QUOTES_`.

Les pods d'application ne se déplient pas initialement après que vous exécutez le script. Le déploiement `famous-quotes` spécifie un sélecteur de nœud et il n'existe pas de nœud de cluster doté d'une étiquette de nœud correspondante.

Supprimez le sélecteur de nœuds du déploiement, ce qui permet à OpenShift de planifier des pods d'application sur n'importe quel nœud disponible.

Créez une route pour l'application, utilisez `quotes.apps.ocp4.example.com` comme nom d'hôte. Vérifiez que l'application répond aux requêtes externes.

## Évaluation

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab review-troubleshoot grade
```

## Fin

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour effectuer cet exercice.

```
[student@workstation ~]$ lab review-troubleshoot finish
```

L'atelier est maintenant terminé.

## ► Solution

# Résoudre les problèmes relatifs aux applications et à un cluster OpenShift

Dans le cadre de cette révision, vous allez permettre aux développeurs d'accéder à un cluster et de résoudre les problèmes liés aux déploiements d'applications.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Crée un projet.
- Effectuer un test de fumée du cluster OpenShift en créant une application à l'aide du processus source-to-image.
- Créer des applications à l'aide de la ressource de déploiement.
- Utiliser le fournisseur d'identité HTPasswd pour gérer les utilisateurs.
- Créer et gérer des groupes.
- Gérer le RBAC et la SCC des groupes et des utilisateurs.
- Gérer les secrets pour les bases de données et les applications.
- Résoudre les problèmes courants.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande :

- Garantit que l'API du cluster est accessible.
- Supprime les utilisateurs et les groupes existants.
- Supprime les fournisseurs d'identité existants.
- Supprime la liaison de rôle de cluster `cluster-admin` de l'utilisateur `admin`.
- Garantit que les utilisateurs authentifiés peuvent créer des projets.

```
[student@workstation ~]$ lab review-troubleshoot start
```

## Instructions

Réalisez les tâches suivantes :

1. En tant qu'utilisateur `kubeadmin`, créez le projet `review-troubleshoot`. Le mot de passe de l'utilisateur `kubeadmin` se trouve dans le fichier `/usr/local/etc/ocp4.config` sur la ligne `RHT_OCP4_KUBEADM_PASSWD`. Effectuez toutes les tâches suivantes dans le projet `review-troubleshoot`.
  - 1.1. Procurez-vous le fichier de configuration de la salle de classe, accessible à l'adresse `/usr/local/etc/ocp4.config` et connectez-vous en tant qu'utilisateur `kubeadmin`.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
[student@workstation ~]$ oc login -u kubeadmin -p ${RHT_OCP4_KUBEADM_PASSWD} \
>     https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

1.2. Créez le projet `review-troubleshoot`.

```
[student@workstation ~]$ oc new-project review-troubleshoot
Now using project "review-troubleshoot" on server
"https://api.ocp4.example.com:6443".
...output omitted...
```

2. Effectuez un test de fumée du cluster pour vérifier la fonctionnalité de base du cluster. Utilisez un `deployment` pour créer une application nommée `hello-world-nginx`. Le code source de l'application se trouve dans le sous-répertoire `hello-world-nginx` du référentiel.

- <https://github.com/RedHatTraining/DO280-apps>

Créez une route pour l'application, utilisez `hello-world.apps.ocp4.example.com` comme nom d'hôte. Vérifiez que l'application répond aux requêtes externes.

2.1. Utilisez la commande `oc new-app` pour créer le déploiement `hello-world-nginx`.

```
[student@workstation ~]$ oc new-app --name hello-world-nginx \
>     https://github.com/RedHatTraining/DO280-apps \
>     --context-dir hello-world-nginx
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "ubi8" created
imagestream.image.openshift.io "hello-world-nginx" created
buildconfig.build.openshift.io "hello-world-nginx" created
deployment.apps "hello-world-nginx" created
service "hello-world-nginx" created
--> Success
...output omitted...
Run 'oc status' to view your app.
```

- 2.2. Créez une route vers l'application en exposant le service `hello-world-nginx`. Définissez le nom d'hôte sur `hello-world.apps.ocp4.example.com`.

```
[student@workstation ~]$ oc expose service hello-world-nginx \
>     --hostname hello-world.apps.ocp4.example.com
route.route.openshift.io/hello-world-nginx exposed
```

2.3. Attendez que le pod d'application soit en cours d'exécution.

```
[student@workstation ~]$ oc get pods
NAME                      READY   STATUS    RESTARTS   AGE
hello-world-nginx-1-build   0/1     Completed  0          2m
hello-world-nginx-695754d9f7-8rv4x  1/1     Running   0          1m
```

2.4. Vérifiez l'accès à l'application.

```
[student@workstation ~]$ curl -s "http://hello-world.apps.ocp4.example.com" | \
>   grep -i Hello
<h1>Hello, world from nginx!</h1>
```

3. Configurez le cluster afin d'utiliser un fournisseur d'identité HTPasswd. Le nom du fournisseur d'identités est `cluster-users`. Le fournisseur d'identités lit les informations d'identification `htpasswd` stockées dans le secret `comprevue-users`. Assurez-vous que quatre comptes d'utilisateurs existent : `admin`, `leader`, `developer` et `qa-engineer`. Tous les comptes d'utilisateurs doivent utiliser `review` comme mot de passe. Ajoutez le rôle `cluster-admin` à l'utilisateur `admin`.

3.1. Créez un fichier d'authentification `htpasswd` temporaire dans `/tmp/cluster-users`.

```
[student@workstation ~]$ touch /tmp/cluster-users
```

3.2. Renseignez le fichier `/tmp/cluster-users` avec les valeurs d'utilisateur et de mot de passe requises.

```
[student@workstation ~]$ for USER in admin leader developer qa-engineer
>   do
>     htpasswd -B -b /tmp/cluster-users "${USER}" review
>   done
Adding password for user admin
Adding password for user leader
Adding password for user developer
Adding password for user qa-engineer
```

3.3. Créez un secret pour `comprevue-users` à partir du fichier `/tmp/cluster-users`.

```
[student@workstation ~]$ oc create secret generic comprevue-users \
>   --from-file htpasswd=/tmp/cluster-users -n openshift-config
secret/comprevue-users created
```

3.4. Exportez la ressource OAuth existante vers un fichier YAML.

```
[student@workstation ~]$ oc get oauth cluster -o yaml > /tmp/oauth.yaml
```

3.5. Modifiez le fichier `/tmp/oauth.yaml` pour ajouter une définition de fournisseur d'identité HTPasswd à la liste `identityProviders`. Définissez le nom du fournisseur d'identité sur `cluster-users` et définissez le nom `fileData` sur `comprevue-users`.

Après avoir apporté ces modifications, le fichier affiche ce qui suit :

```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
  ...output omitted...
  name: cluster
  ...output omitted...
spec:
  identityProviders:
    - name: cluster-users
      mappingMethod: claim
      type: HTPasswd
      htpasswd:
        fileData:
          name: compreview-users
```



#### Note

Les clés `name`, `mappingMethod`, `type` et `htpasswd` utilisent toutes la même mise en retrait.

- 3.6. Remplacez la ressource OAuth existante par la définition de ressource dans le fichier modifié :

```
[student@workstation ~]$ oc replace -f /tmp/oauth.yaml
oauth.config.openshift.io/cluster replaced
```

- 3.7. Assignez le rôle `admin` à l'utilisateur `cluster-admin`.

```
[student@workstation ~]$ oc adm policy add-cluster-role-to-user \
>   cluster-admin admin
Warning: User 'admin' not found
clusterrole.rbac.authorization.k8s.io/cluster-admin added: "admin"
```



#### Note

Vous pouvez ignorer l'avertissement en toute sécurité concernant l'utilisateur `admin` qui est introuvable. La ressource `user/admin` n'existe pas dans votre cluster OpenShift tant que l'utilisateur `admin` n'est pas connecté pour la première fois.

- 3.8. Attendez que les pods de l'espace de noms `openshift-authentication` soient redémarrés.

```
[student@workstation ~]$ oc get pods -n openshift-authentication
NAME                  READY   STATUS    RESTARTS   AGE
oauth-openshift-768c6476d4-tg8kc  1/1     Running   0          40s
oauth-openshift-768c6476d4-9dc48  1/1     Running   0          60s
oauth-openshift-768c6476d4-qngxw  1/1     Running   0          80s
```

**Note**

Vous devrez peut-être exécuter la commande plusieurs fois jusqu'à ce que la condition souhaitée soit atteinte.

4. En tant qu'utilisateur `admin`, créez trois groupes d'utilisateurs : `leaders`, `developers` et `qa`.

Affectez l'utilisateur `leader` au groupe `leaders`, l'utilisateur `developer` aux groupes `developers` et l'utilisateur `qa-engineer` au groupe `qa`.

Affectez des rôles à chaque groupe :

- Affectez le rôle `self-provisioner` au groupe `leaders`, ce qui permet aux membres de créer des projets. Pour que ce rôle soit efficace, vous devez également supprimer la capacité de tout utilisateur authentifié à créer de nouveaux projets.
- Affectez le rôle `edit` au groupe `developers` pour le projet `review-troubleshoot` uniquement, ce qui permet aux membres de créer et de supprimer des ressources de projet.
- Affectez le rôle `view` au groupe `qa` pour le projet `review-troubleshoot` uniquement, ce qui offre aux membres un accès en lecture aux ressources de projet.

4.1. Connectez-vous en tant qu'utilisateur `admin`.

```
[student@workstation ~]$ oc login -u admin -p review
Login successful.

...output omitted...
```

**Note**

Si la commande de connexion échoue, vérifiez que les pods de l'espace de noms `openshift-authentication` sont redémarrés et en cours d'exécution.

4.2. Créez les trois groupes d'utilisateurs.

```
[student@workstation ~]$ for GROUP in leaders developers qa
> do
>   oc adm groups new "${GROUP}"
> done
group.user.openshift.io/leaders created
group.user.openshift.io/developers created
group.user.openshift.io/qa created
```

4.3. Ajoutez chaque utilisateur au groupe approprié.

```
[student@workstation ~]$ oc adm groups add-users leaders leader
group.user.openshift.io/leaders added: "leader"

[student@workstation ~]$ oc adm groups add-users developers developer
group.user.openshift.io/developers added: "developer"

[student@workstation ~]$ oc adm groups add-users qa qa-engineer
group.user.openshift.io/qa added: "qa-engineer"
```

4.4. Autoriser les membres du groupe **leaders** à créer de nouveaux projets :

```
[student@workstation ~]$ oc adm policy add-cluster-role-to-group \
>   self-provisioner leaders
clusterrole.rbac.authorization.k8s.io/self-provisioner added: "leaders"
```

4.5. Supprimez le rôle de cluster **self-provisioner** du groupe **system:authenticated:oauth**.

```
[student@workstation ~]$ oc adm policy remove-cluster-role-from-group \
>   self-provisioner system:authenticated:oauth
Warning: Your changes may get lost whenever a master is restarted, unless you
prevent reconciliation of this rolebinding using the following command:
oc annotate clusterrolebinding.rbac self-provisioners
'rbac.authorization.kubernetes.io/autoupdate=false' --overwrite
clusterrole.rbac.authorization.k8s.io/self-provisioner removed:
"system:authenticated:oauth"
```



### Note

Vous pouvez ignorer en toute sécurité l'avertissement indiquant que vos modifications seront perdues.

4.6. Autorisez les membres du groupe **developers** à créer et à supprimer des ressources dans le projet **review-troubleshoot** :

```
[student@workstation ~]$ oc policy add-role-to-group edit developers
clusterrole.rbac.authorization.k8s.io/edit added: "developers"
```

4.7. Autorisez les membres du groupe **qa** à visualiser les ressources du projet :

```
[student@workstation ~]$ oc policy add-role-to-group view qa
clusterrole.rbac.authorization.k8s.io/view added: "qa"
```

5. En tant qu'utilisateur **developer**, utilisez un déploiement pour créer une application nommée **mysql** dans le projet **review-troubleshoot**. Utilisez l'image disponible dans **registry.redhat.io/rhel8/mysql-80:1-139**. Cette application fournit un service de base de données partagé pour d'autres applications de projet.

Créez un secret générique appelé **mysql** avec **password** comme clé et **r3dh4t123** comme valeur.

Définissez les variables d'environnement **MYSQL\_ROOT\_** à partir des valeurs du secret **mysql**.

**chapitre 16 |** Révision complète

Configurez l'application de base de données mysql pour monter une revendication de volume persistant (PVC) dans le répertoire /var/lib/mysql/data à l'intérieur du pod. La PVC doit avoir une taille de 2 GB et ne doit demander que le mode d'accès ReadWriteOnce.

- 5.1. Connectez-vous au cluster en tant qu'utilisateur developer.

```
[student@workstation ~]$ oc login -u developer -p review
Login successful.

...output omitted...
```

- 5.2. Créez une application pour déployer un serveur de base de données mysql. Utilisez la commande oc new-app pour créer un déploiement.

```
[student@workstation ~]$ oc new-app --name mysql \
>   --image registry.redhat.io/rhel8/mysql-80:1-139
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "mysql" created
  deployment.apps "mysql" created
  service "mysql" created
--> Success
...output omitted...
Run 'oc status' to view your app.
```

- 5.3. Créez un secret générique pour la base de données MySQL appelée mysql avec password comme clé et r3dh4t123 comme valeur.

```
[student@workstation ~]$ oc create secret generic mysql \
>   --from-literal password=r3dh4t123
secret/mysql created
```

**Note**

Vous allez utiliser ce secret pour fournir le mot de passe de la base de données pour l'application.

- 5.4. Utilisez le secret mysql afin d'initialiser les variables d'environnement pour le déploiement mysql.

```
[student@workstation ~]$ oc set env deployment mysql \
>   --prefix MYSQL_ROOT_ --from secret/mysql
deployment.apps/mysql updated
```

- 5.5. Utilisez la commande oc set volumes afin de configurer le stockage persistant pour le déploiement mysql. La commande crée automatiquement une revendication de volume persistant avec la taille et le mode d'accès spécifiés. Le montage du volume dans le répertoire /var/lib/mysql/data permet d'accéder aux données stockées, même si un pod de base de données est supprimé et qu'un autre est créé.

```
[student@workstation ~]$ oc set volumes deployment/mysql --name mysql-storage \
>   --add --type pvc --claim-size 2Gi --claim-mode rwo \
>   --mount-path /var/lib/mysql/data
deployment.apps/mysql volume updated
```

- 5.6. Vérifiez que le pod `mysql` a bien été redéployé après avoir configuré le déploiement pour qu'il utilise un secret et un volume.

```
[student@workstation ~]$ oc get pods -l deployment=mysql
NAME           READY   STATUS    RESTARTS   AGE
mysql-bbb6b5fbb-dmq9x   1/1     Running   0          60s
```



### Note

Vous devrez peut-être exécuter plusieurs fois la commande jusqu'à ce que le pod `mysql` affiche `1/1` et `Running`.

- 5.7. Vérifiez qu'il existe une revendication de volume persistant avec la taille et le mode d'accès corrects.

```
[student@workstation ~]$ oc get pvc
NAME      STATUS  ...  CAPACITY  ACCESS MODES  STORAGECLASS  AGE
pvc-ks52v  Bound   ...  2Gi       RWO          nfs-storage   2m33s
```

- 5.8. Vérifiez que le pod `mysql` en cours d'exécution monte un volume dans le répertoire `/var/lib/mysql/data`.

```
[student@workstation ~]$ oc exec -it mysql-bbb6b5fbb-dmq9x -- \
>   df -h /var/lib/mysql/data
Filesystem           Size  Used  Avail Use% Mounted on
...:/exports/review-troubleshoot-pvc-...  40G  859M   40G   3% /var/lib/mysql/data
```

6. En tant qu'utilisateur `developer`, utilisez un `deployment` pour créer une application nommée `wordpress`. Créez l'application dans le projet `review-troubleshoot`. Utilisez l'image disponible dans `quay.io/redhattraining/wordpress:5.7-php7.4-apache`. L'application Wordpress nécessite la définition de plusieurs variables d'environnement lors de l'exécution de la commande `oc new-app` :

- `WORDPRESS_DB_HOST` avec la valeur `mysql`.
- `WORDPRESS_DB_NAME` pour avoir la valeur `wordpress`.
- `WORDPRESS_DB_USER` pour avoir la valeur `root`.
- `WORDPRESS_USER` avec la valeur `wpuser`.
- `WORDPRESS_PASSWORD` avec la valeur `wppass`.
- `WORDPRESS_TITLE` pour avoir la valeur `review-troubleshoot`.
- `WORDPRESS_URL` avec la valeur `wordpress.apps.ocp4.example.com`.
- `WORDPRESS_EMAIL` avec la valeur `student@redhat.com`.

Définissez les variables d'environnement `WORDPRESS_DB_*` pour récupérer le mot de passe de la base de données à partir du secret `mysql`.

L'application `wordpress` requiert la contrainte de contexte de sécurité `anyuid`. Créez un compte de service nommé `wordpress-sa`, puis attribuez-lui la contrainte de contexte de

**chapitre 16 |** Révision complète

sécurité anyuid. Configurez le déploiement wordpress pour utiliser le compte de service wordpress-sa.

L'application wordpress requiert également l'existence de la base de données WORDPRESS\_DB\_NAME sur le serveur de base de données. Créez une base de données vide nommée wordpress en exécutant la commande SQL CREATE DATABASE wordpress dans le pod MySQL.

Créez une route pour l'application, utilisez wordpress.apps.ocp4.example.com comme nom d'hôte. Si vous déployez correctement l'application, un assistant d'installation s'affiche lorsque vous accédez à l'application depuis un navigateur.

6.1. Déployez une application wordpress en tant que déploiement.

```
[student@workstation ~]$ oc new-app --name wordpress \
>   --image quay.io/redhattraining/wordpress:5.7-php7.4-apache \
>   -e WORDPRESS_DB_HOST=mysql \
>   -e WORDPRESS_DB_NAME=wordpress \
>   -e WORDPRESS_DB_USER=root \
>   -e WORDPRESS_USER=wpuser \
>   -e WORDPRESS_PASSWORD=wppass \
>   -e WORDPRESS_TITLE=review-troubleshoot \
>   -e WORDPRESS_URL=wordpress.apps.ocp4.example.com \
>   -e WORDPRESS_EMAIL=student@redhat.com
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "wordpress" created
  deployment.apps "wordpress" created
  service "wordpress" created
--> Success
...output omitted...
Run 'oc status' to view your app.
```

6.2. Ajoutez les variables d'environnement WORDPRESS\_DB\_\* au déploiement wordpress. Le secret contient le mot de passe de la base de données.

```
[student@workstation ~]$ oc set env deployment/wordpress \
>   --prefix WORDPRESS_DB_ --from secret/mysql
deployment.apps/wordpress updated
```

6.3. Créez le compte de service wordpress-sa.

```
[student@workstation ~]$ oc create serviceaccount wordpress-sa
serviceaccount/wordpress-sa created
```

6.4. Connectez-vous au cluster en tant qu'utilisateur admin et accordez les priviléges anyuid au compte de service wordpress-sa.

```
[student@workstation ~]$ oc login -u admin -p review
Login successful.

...output omitted...
[student@workstation ~]$ oc adm policy add-scc-to-user anyuid -z wordpress-sa
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:anyuid added:
"wordpress-sa"
```

- 6.5. Revenez à l'utilisateur developer pour effectuer les étapes restantes. Connectez-vous au cluster en tant qu'utilisateur developer.

```
[student@workstation ~]$ oc login -u developer -p review
Login successful.

...output omitted...
```

- 6.6. Configurez le déploiement wordpress pour utiliser le compte de service wordpress-sa.

```
[student@workstation ~]$ oc set serviceaccount deployment/wordpress \
>   wordpress-sa
deployment.apps/wordpress serviceaccount updated
```

- 6.7. Créez la base de données wordpress. Utilisez l'outil client mysql sur le pod en cours d'exécution.

```
[student@workstation ~]$ oc get pods -l deployment=mysql
NAME          READY   STATUS    RESTARTS   AGE
mysql-fbf67ff96-c4sq7   1/1     Running   0          10m

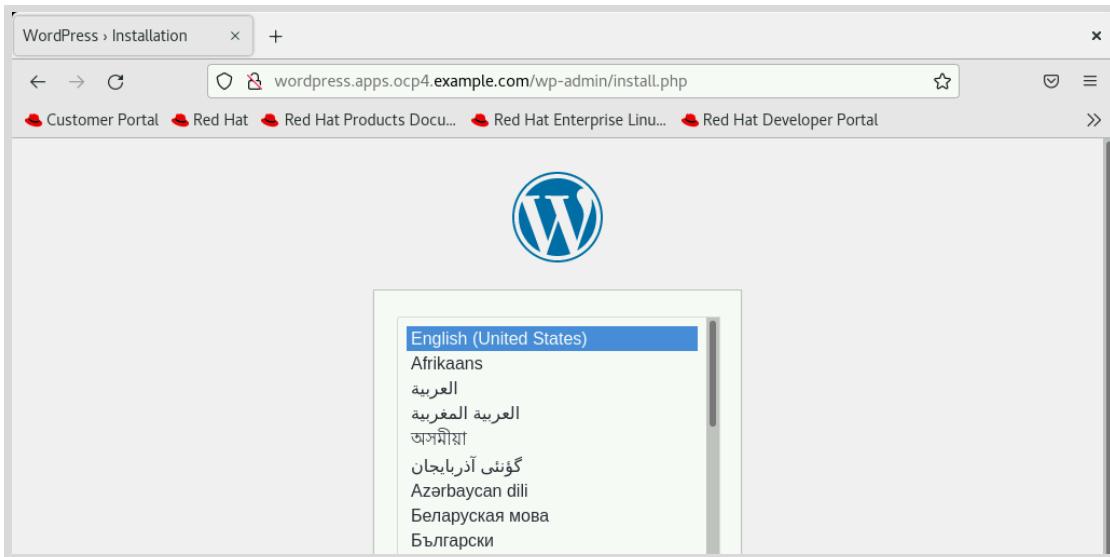
[student@workstation ~]$ oc exec mysql-fbf67ff96-c4sq7 -- \
>   /usr/bin/mysql -uroot -e "CREATE DATABASE wordpress"
```

- 6.8. Créez une route pour l'application wordpress. Définissez le nom d'hôte sur wordpress.apps.ocp4.example.com.

```
[student@workstation ~]$ oc expose service wordpress \
>   --hostname wordpress.apps.ocp4.example.com
route.route.openshift.io/wordpress exposed
```

- 6.9. Utilisez un navigateur Web pour vérifier l'accès à l'URL de l'application. Lorsque vous déployez correctement l'application, un assistant d'installation s'affiche dans le navigateur.

- <http://wordpress.apps.ocp4.example.com>



7. En tant qu'utilisateur developer, déployez l'application famous-quotes dans le projet review-troubleshoot à l'aide du script ~/D0280/labs/review-troubleshoot/deploy\_famous-quotes.sh. Ce script crée la base de données defaultdb et les ressources définies dans le fichier ~/D0280/labs/review-troubleshoot/famous-quotes.yaml.

Utilisez le secret mysql afin d'initialiser les variables d'environnement pour le déploiement famous-quotes avec le préfixe QUOTES\_.

Les pods d'application ne se déplient pas initialement après que vous exécutez le script. Le déploiement famous-quotes spécifie un sélecteur de nœud et il n'existe pas de nœud de cluster doté d'une étiquette de nœud correspondante.

Supprimez le sélecteur de nœuds du déploiement, ce qui permet à OpenShift de planifier des pods d'application sur n'importe quel nœud disponible.

Créez une route pour l'application, utilisez quotes.apps.ocp4.example.com comme nom d'hôte. Vérifiez que l'application répond aux requêtes externes.

- 7.1. Exécutez le script ~/D0280/labs/review-troubleshoot/deploy\_famous-quotes.sh.

```
[student@workstation ~]$ ~/D0280/labs/review-troubleshoot/deploy_famous-quotes.sh
Creating famous-quotes database
Deploying famous-quotes application
deployment.apps/famous-quotes created
service/famous-quotes created
```

- 7.2. Utilisez le secret mysql afin d'initialiser les variables d'environnement avec le préfixe QUOTES\_.

```
[student@workstation ~]$ oc set env deployment famous-quotes \
> --prefix QUOTES_ --from secret/mysql
```

- 7.3. Vérifiez que le pod d'application famous-quotes n'est pas planifié pour déploiement.

```
[student@workstation ~]$ oc get pods -l app=famous-quotes
NAME                  READY   STATUS    RESTARTS   AGE
famous-quotes-85ff8679d7-vhvbk   0/1     Pending   0          41s
```

- 7.4. Vérifiez si des événements de projet contiennent des informations sur le problème.

```
[student@workstation ~]$ oc get events --sort-by='(.lastTimestamp)' | \
> grep -i Scheduling
...output omitted...
30s  Warning  FailedScheduling      pod/famous-quotes-1-deploy      0/3 nodes are
available: 3 node(s) didn't match node selector.
...output omitted...
```

- 7.5. Enregistrez la ressource de déploiement famous-quotes dans un fichier.

```
[student@workstation ~]$ oc get deployment/famous-quotes \
> -o yaml > /tmp/famous-deploy.yaml
```

- 7.6. Utilisez un éditeur pour supprimer le sélecteur de nœud du fichier /tmp/famous-deploy.yaml. Recherchez la ligne `nodeSelector` dans le fichier. Supprimez ensuite les deux lignes suivantes du fichier /tmp/famous-deploy.yaml.

```
nodeSelector:
  env: quotes
```

- 7.7. Remplacez le déploiement famous-quotes par le fichier modifié.

```
[student@workstation ~]$ oc replace -f /tmp/famous-deploy.yaml
deployment.apps/famous-quotes replaced
```

- 7.8. Attendez quelques instants et exécutez la commande `oc get pods` afin de vous assurer que l'application famous-quotes est désormais en cours d'exécution.

```
[student@workstation ~]$ oc get pods -l app=famous-quotes
NAME                  READY   STATUS    RESTARTS   AGE
famous-quotes-2-gmzzj   1/1     Running   0          53s
```



### Note

Vous devrez peut-être exécuter la commande plusieurs fois jusqu'à ce que la condition souhaitée soit atteinte.

- 7.9. Créez une route pour l'application famous-quotes. Définissez le nom d'hôte sur `quotes.apps.ocp4.example.com`.

```
[student@workstation ~]$ oc expose service famous-quotes \
> --hostname quotes.apps.ocp4.example.com
route.route.openshift.io/famous-quotes exposed
```

7.10. Vérifiez que l'application famous-quotes répond aux demandes envoyées à l'URL.

- `http://quotes.apps.ocp4.example.com`

```
[student@workstation ~]$ curl -s "http://quotes.apps.ocp4.example.com" | \
>   egrep '</?title>' \
<title>Quotes</title>
```

## Évaluation

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab review-troubleshoot grade
```

## Fin

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour effectuer cet exercice.

```
[student@workstation ~]$ lab review-troubleshoot finish
```

L'atelier est maintenant terminé.

## ► Open Lab

# Configurer un modèle de projet avec des restrictions de ressources et de réseau

Dans le cadre de cette révision, vous allez configurer le modèle de projet par défaut du cluster pour vous assurer que les nouveaux projets appliquent les quotas, les limites de ressources et les politiques réseau par défaut.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Modifier le modèle de projet par défaut pour créer automatiquement des plages de limites, des quotas de ressources et des politiques réseau.
- Créer un secret TLS à l'aide des fichiers fournis.
- Monter un secret en tant que volume au sein d'une application.
- Créer une route directe vers une application.
- Configurer une application pour qu'elle soit automatiquement mise à l'échelle.

## Avant De Commencer

En tant qu'utilisateur student sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande :

- Garantit que l'API du cluster est accessible.
- Configure le fournisseur d'identité HTPasswd et fournit l'accès aux utilisateurs `admin`, `leader` et `developer`.
- Télécharge des exemples de fichiers YAML dans `~/D0280/labs/review-template/sample-files`.
- Crée le projet `review-template-test` et déploie une application dans le projet que vous pouvez utiliser pour tester vos politiques réseau.
- Ajoute le libellé `network.openshift.io/policy-group=ingress` à l'espace de noms `default`.
- Génère les fichiers de certificat dont l'application sécurisée a besoin.

```
[student@workstation ~]$ lab review-template start
```

**Note**

Si vous avez besoin d'aide pour commencer, consultez le chapitre *Post-installation network configuration* du guide de configuration de la post-installation de Red Hat OpenShift Container Platform.

## Instructions

Réalisez les tâches suivantes :

1. En tant qu'utilisateur `admin`, mettez à jour le cluster OpenShift de manière à utiliser un nouveau modèle de projet. Le modèle de projet doit créer automatiquement la politique réseau, la plage de limites et ajouter des ressources de quota pour les nouveaux projets. Les nouveaux projets doivent automatiquement recevoir un libellé correspondant au nom du projet. Par exemple, un projet nommé `test` possède le libellé `name=test`.

Le tableau suivant vous guide vers les ressources nécessaires.

Ressource	Exigences
Project	<ul style="list-style-type: none"> <li>Inclut un libellé avec le nom du projet.</li> </ul>
NetworkPolicy	<p><b>Policy 1:</b></p> <ul style="list-style-type: none"> <li>Les routes sont accessibles au trafic externe ; cela signifie que le trafic est autorisé en provenance des pods dans les espaces de noms avec le libellé <code>network.openshift.io/policy-group=ingress</code>.</li> </ul> <p><b>Policy 2:</b></p> <ul style="list-style-type: none"> <li>Les pods du même espace de noms peuvent communiquer entre eux.</li> <li>Les pods ne répondent pas à ceux qui figurent dans un espace de noms différent, à l'exception des espaces de noms ayant le libellé <code>network.openshift.io/policy-group=ingress</code>.</li> </ul>
LimitRange	<ul style="list-style-type: none"> <li>Chaque conteneur demande 30 millicores de processeur.</li> <li>Chaque conteneur demande 30 Mio de mémoire.</li> <li>Chaque conteneur est limité à 100 millicores de processeur.</li> <li>Chaque conteneur est limité à 100 Mio de mémoire.</li> </ul>
ResourceQuota	<ul style="list-style-type: none"> <li>Les projets sont limités à 10 pods.</li> <li>Les projets peuvent demander un maximum de 1 Gio de mémoire.</li> <li>Les projets peuvent demander un maximum de 2 processeurs.</li> <li>Les projets peuvent utiliser un maximum de 4 Gio de mémoire.</li> <li>Les projets peuvent utiliser un maximum de 4 processeurs.</li> </ul>

2. En tant qu'utilisateur `developer`, créez un projet nommé `review-template`. Assurez-vous que le projet `review-template` hérite des paramètres spécifiés dans le nouveau modèle de projet. Dans le projet `review-template`, créez un déploiement nommé `hello-secure`

à l'aide de l'image de conteneur située dans quay.io/redhattraining/hello-world-secure:v1.0.

**Note**

Le pod hello-secure ne s'exécute pas correctement tant que vous n'avez pas fourni l'accès au certificat TLS et à la clé requis par le serveur NGINX.

3. En tant qu'utilisateur developer, créez un secret TLS à l'aide du certificat hello-secure-combined.pem et de la clé hello-secure-key.pem situés dans le répertoire ~/D0280/labs/review-template. Utilisez les journaux du pod hello-secure qui a échoué pour déterminer le point de montage attendu pour le certificat. Montez le secret TLS en tant que volume dans le pod à l'aide du répertoire identifié. Vérifiez que le pod hello-secure est redéployé correctement.
4. Le certificat hello-secure-combined.pem est valide pour un seul nom d'hôte. Utilisez la commande openssl x509 avec les options -noout et -ext 'subjectAltName' pour lire le certificat hello-secure-combined.pem et identifier le nom d'hôte. En tant qu'utilisateur developer, créez une route directe vers le service hello-secure à l'aide du nom d'hôte identifié. Vérifiez que la route répond aux requêtes externes.

**Note**

La page man x509(1) fournit des informations sur la commande openssl x509.

5. En tant qu'utilisateur developer, configurez le déploiement hello-secure pour qu'il soit mis à l'échelle automatiquement. Il doit y avoir au moins un pod en cours d'exécution dans le déploiement. Si l'utilisation moyenne du processeur est supérieure à 80 %, le déploiement augmente sur un maximum de cinq pods.

**Note**

Vous pouvez utiliser le script situé dans ~/D0280/solutions/review-template/test-hpa.sh pour vérifier que votre déploiement est mis à l'échelle comme prévu.

## Évaluation

En tant qu'utilisateur student sur la machine workstation, utilisez la commande lab pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab review-template grade
```

## Fin

En tant qu'utilisateur student sur la machine workstation, utilisez la commande lab pour effectuer cet exercice.

```
[student@workstation ~]$ lab review-template finish
```

L'atelier est maintenant terminé.

## ► Solution

# Configurer un modèle de projet avec des restrictions de ressources et de réseau

Dans le cadre de cette révision, vous allez configurer le modèle de projet par défaut du cluster pour vous assurer que les nouveaux projets appliquent les quotas, les limites de ressources et les politiques réseau par défaut.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Modifier le modèle de projet par défaut pour créer automatiquement des plages de limites, des quotas de ressources et des politiques réseau.
- Créer un secret TLS à l'aide des fichiers fournis.
- Monter un secret en tant que volume au sein d'une application.
- Créer une route directe vers une application.
- Configurer une application pour qu'elle soit automatiquement mise à l'échelle.

## Avant De Commencer

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` en vue de préparer votre système pour cet exercice.

La commande :

- Garantit que l'API du cluster est accessible.
- Configure le fournisseur d'identité HTPasswd et fournit l'accès aux utilisateurs `admin`, `leader` et `developer`.
- Télécharge des exemples de fichiers YAML dans `~/D0280/labs/review-template/sample-files`.
- Crée le projet `review-template-test` et déploie une application dans le projet que vous pouvez utiliser pour tester vos politiques réseau.
- Ajoute le libellé `network.openshift.io/policy-group=ingress` à l'espace de noms `default`.
- Génère les fichiers de certificat dont l'application sécurisée a besoin.

```
[student@workstation ~]$ lab review-template start
```

**Note**

Si vous avez besoin d'aide pour commencer, consultez le chapitre *Post-installation network configuration* du guide de configuration de la post-installation de Red Hat OpenShift Container Platform.

## Instructions

Réalisez les tâches suivantes :

1. En tant qu'utilisateur **admin**, mettez à jour le cluster OpenShift de manière à utiliser un nouveau modèle de projet. Le modèle de projet doit créer automatiquement la politique réseau, la plage de limites et ajouter des ressources de quota pour les nouveaux projets. Les nouveaux projets doivent automatiquement recevoir un libellé correspondant au nom du projet. Par exemple, un projet nommé `test` possède le libellé `name=test`.

Le tableau suivant vous guide vers les ressources nécessaires.

Ressource	Exigences
Project	<ul style="list-style-type: none"> <li>Inclut un libellé avec le nom du projet.</li> </ul>
NetworkPolicy	<p><b>Policy 1:</b></p> <ul style="list-style-type: none"> <li>Les routes sont accessibles au trafic externe ; cela signifie que le trafic est autorisé en provenance des pods dans les espaces de noms avec le libellé <code>network.openshift.io/policy-group=ingress</code>.</li> </ul> <p><b>Policy 2:</b></p> <ul style="list-style-type: none"> <li>Les pods du même espace de noms peuvent communiquer entre eux.</li> <li>Les pods ne répondent pas à ceux qui figurent dans un espace de noms différent, à l'exception des espaces de noms ayant le libellé <code>network.openshift.io/policy-group=ingress</code>.</li> </ul>
LimitRange	<ul style="list-style-type: none"> <li>Chaque conteneur demande 30 millicores de processeur.</li> <li>Chaque conteneur demande 30 Mio de mémoire.</li> <li>Chaque conteneur est limité à 100 millicores de processeur.</li> <li>Chaque conteneur est limité à 100 Mio de mémoire.</li> </ul>
ResourceQuota	<ul style="list-style-type: none"> <li>Les projets sont limités à 10 pods.</li> <li>Les projets peuvent demander un maximum de 1 Gio de mémoire.</li> <li>Les projets peuvent demander un maximum de 2 processeurs.</li> <li>Les projets peuvent utiliser un maximum de 4 Gio de mémoire.</li> <li>Les projets peuvent utiliser un maximum de 4 processeurs.</li> </ul>

- 1.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur **admin**.

```
[student@workstation ~]$ oc login -u admin -p redhat \
>   https://api.ocp4.example.com:6443
Login successful.

...output omitted...
```

- 1.2. Utilisez la commande `oc adm create-bootstrap-project-template` pour créer un fichier YAML que vous allez personnaliser pour cet exercice. Enregistrez le fichier dans `~/D0280/labs/review-template/project-template.yaml`.

```
[student@workstation ~]$ oc adm create-bootstrap-project-template \
>   -o yaml > ~/D0280/labs/review-template/project-template.yaml
```

- 1.3. Choisissez le répertoire `~/D0280/labs/review-template`.

```
[student@workstation ~]$ cd ~/D0280/labs/review-template
```

- 1.4. Modifiez le fichier `project-template.yaml` afin d'ajouter un libellé pour les nouveaux projets. Ajoutez les lignes en gras, en veillant à ce que la mise en retrait soit correcte, puis enregistrez le fichier. La variable `PROJECT_NAME` prend la valeur du nom du projet.

```
...output omitted...
- apiVersion: project.openshift.io/v1
  kind: Project
  metadata:
    labels:
      name: ${PROJECT_NAME}
    annotations:
...output omitted...
```

- 1.5. Listez les fichiers dans le répertoire `~/D0280/labs/review-template/sample-files`. Le répertoire fournit deux exemples de fichiers de politiques réseau, un exemple de fichier de plages de limites et un exemple de fichier de quotas de ressources.

```
[student@workstation review-template]$ ls sample-files
allow-from-openshift-ingress.yaml      limitrange.yaml
allow-same-namespace.yaml              resourcequota.yaml
```

- 1.6. Ajoutez le contenu des fichiers `sample-files/allow-*.yaml` au fichier `project-template.yaml`. Étant donné que le fichier `project-template.yaml` attend une liste de ressources, la première ligne de chaque ressource doit commencer par `-` et vous devez mettre en retrait le reste du contenu en conséquence.  
Modifiez le fichier `project-template.yaml` pour ajouter les ressources de la politique réseau. Ajoutez les lignes en gras, en veillant à ce que la mise en retrait soit correcte, puis enregistrez le fichier.

```

...output omitted...
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: ${PROJECT_ADMIN_USER}
- apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
    name: allow-from-openshift-ingress
  spec:
    podSelector: {}
    ingress:
      - from:
          - namespaceSelector:
              matchLabels:
                network.openshift.io/policy-group: ingress
- apiVersion: networking.k8s.io/v1
  kind: NetworkPolicy
  metadata:
    name: allow-same-namespace
  spec:
    podSelector: {}
    ingress:
      - from:
          - podSelector: {}
parameters:
- name: PROJECT_NAME
...output omitted...

```

- 1.7. Ajoutez le contenu des fichiers sample-files/limitrange.yaml et sample-files/resourcequota.yaml au fichier project-template.yaml. Étant donné que le fichier project-template.yaml attend une liste de ressources, la première ligne de chaque ressource doit commencer par - et vous devez mettre en retrait le reste du contenu en conséquence.

Modifiez le fichier project-template.yaml pour ajouter les ressources de quota et la plage de limites. Ajoutez les lignes en gras, en veillant à ce que la mise en retrait soit correcte, puis enregistrez le fichier.

```

...output omitted...
ingress:
- from:
  - podSelector: {}
- apiVersion: v1
  kind: LimitRange
  metadata:
    name: project-limitrange
  spec:
    limits:
      - default:
          memory: 100Mi
          cpu: 100m
    defaultRequest:

```

```
memory: 30Mi
cpu: 30m
type: Container
- apiVersion: v1
  kind: ResourceQuota
  metadata:
    name: project-quota
  spec:
    hard:
      pods: '10'
      requests.cpu: '2'
      requests.memory: 1Gi
      limits.cpu: '4'
      limits.memory: 4Gi
  parameters:
- name: PROJECT_NAME
...output omitted...
```

- 1.8. Utilisez la commande `oc create` pour créer une ressource de modèle dans l'espace de noms `openshift-config` à l'aide du fichier `project-template.yaml`.

**Note**

Le fichier `~/D0280/solutions/review-template/project-template.yaml` contient la configuration correcte et peut être utilisé à des fins de comparaison.

```
[student@workstation review-template]$ oc create -f project-template.yaml \
> -n openshift-config
template.template.openshift.io/project-request created
```

- 1.9. Listez les modèles dans l'espace de noms `openshift-config`. Vous utiliserez le nom du modèle à l'étape suivante.

```
[student@workstation review-template]$ oc get templates -n openshift-config
NAME          DESCRIPTION      PARAMETERS      OBJECTS
project-request           5 (5 blank)     6
```

- 1.10. Mettez à jour la ressource `projects.config.openshift.io/cluster` pour utiliser le nouveau modèle.

```
[student@workstation review-template]$ oc edit \
> projects.config.openshift.io/cluster
```

Modifiez la ligne `spec: {}` pour qu'elle corresponde aux lignes en gras ci-dessous. Utilisez le nom de modèle identifié dans l'espace de noms `openshift-config`. Assurez-vous que la mise en retrait est correcte, puis enregistrez vos modifications.

```
...output omitted...
spec:
  projectRequestTemplate:
    name: project-request
```

- 1.11. Après une modification réussie, les pods `apiserver` de l'espace de noms `openshift-apiserver` sont redéployés. Surveillez le redéploiement.

```
[student@workstation review-template]$ watch oc get pods -n openshift-apiserver
```

Appuyez sur `Ctrl+C` pour mettre fin à la commande `watch` une fois que les trois nouveaux pods sont en cours d'exécution.

```
Every 2.0s: oc get pods -n openshift-apiserver      ...
NAME          READY   STATUS    RESTARTS   AGE
apiserver-75cfdfc877-257vs  2/2     Running   0          61s
apiserver-75cfdfc877-l2xnv  2/2     Running   0          29s
apiserver-75cfdfc877-rn9fs  2/2     Running   0          47s
```

2. En tant qu'utilisateur `developer`, créez un projet nommé `review-template`. Assurez-vous que le projet `review-template` hérite des paramètres spécifiés dans le nouveau modèle de projet. Dans le projet `review-template`, créez un déploiement nommé `hello-secure` à l'aide de l'image de conteneur située dans `quay.io/redhattraining/hello-world-secure:v1.0`.



#### Note

Le pod `hello-secure` ne s'exécute pas correctement tant que vous n'avez pas fourni l'accès au certificat TLS et à la clé requis par le serveur NGINX.

- 2.1. Connectez-vous à votre cluster OpenShift en tant qu'utilisateur `developer`.

```
[student@workstation review-template]$ oc login -u developer -p developer
Login successful.
```

```
...output omitted...
```

- 2.2. Créez le projet `review-template`.

```
[student@workstation review-template]$ oc new-project review-template
Now using project "review-template" on server "https://api.ocp4.example.com:6443".
...output omitted...
```

- 2.3. Listez la politique réseau, la plage de limites et les ressources de quota du projet `review-template`.

```
[student@workstation review-template]$ oc get networkpolicies
NAME          POD-SELECTOR   AGE
allow-from-openshift-ingress <none>       94s
```

**chapitre 16 |** Révision complète

```
allow-same-namespace <none> 94s

[student@workstation review-template]$ oc get limitranges
NAME          CREATED AT
project-limitrange  2022-05-04T21:53:09Z

[student@workstation review-template]$ oc get resourcequotas
NAME      AGE     REQUEST
project-quota  103s   pods: 0/10, requests.cpu: 0/2, requests.memory: 0/1Gi

                                LIMIT
                                limits.cpu: 0/4, limits.memory: 0/4Gi
```

- 2.4. Vérifiez que le libellé `review-template` est associé au projet `name=review-template`.

```
[student@workstation review-template]$ oc get project review-template \
> --show-labels
NAME          DISPLAY NAME  STATUS  LABELS
review-template           Active  ... ,name=review-template
```

- 2.5. Utilisez la commande `oc new-app` pour créer le déploiement `hello-secure` à l'aide de l'image de conteneur `quay.io/redhattraining/hello-world-secure:v1.0`.

```
[student@workstation review-template]$ oc new-app --name hello-secure \
> --image quay.io/redhattraining/hello-world-secure:v1.0
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "hello-secure" created
  deployment.apps "hello-secure" created
  service "hello-secure" created
--> Success
...output omitted...
Run 'oc status' to view your app.
```

- 2.6. Vérifiez que le pod `hello-secure` ne démarre pas correctement.

```
[student@workstation review-template]$ watch oc get pods
```

Appuyez sur `Ctrl+C` pour mettre fin à la commande `watch` une fois que l'état du pod est `CrashLoopBackOff` ou `Error`.

```
Every 2.0s: oc get pods               ...

NAME          READY  STATUS        RESTARTS  AGE
hello-secure-6475f657c9-rmgsr  0/1    CrashLoopBackOff  1          14s
```

3. En tant qu'utilisateur `developer`, créez un secret TLS à l'aide du certificat `hello-secure-combined.pem` et de la clé `hello-secure-key.pem` situés dans le répertoire `~/DO280/labs/review-template`. Utilisez les journaux du pod `hello-secure` qui a échoué pour déterminer le point de montage attendu pour le certificat. Montez le secret TLS en tant que

**chapitre 16 |** Révision complète

volume dans le pod à l'aide du répertoire identifié. Vérifiez que le pod `hello-secure` est redéployé correctement.

- 3.1. Créez un secret TLS à l'aide du certificat `hello-secure-combined.pem` et de la clé `hello-secure-key.pem`.

```
[student@workstation review-template]$ oc create secret tls hello-tls \
>   --cert hello-secure-combined.pem --key hello-secure-key.pem
secret/hello-tls created
```

- 3.2. Identifiez le nom du pod qui a échoué.

```
[student@workstation review-template]$ oc get pods
NAME                  READY   STATUS        RESTARTS   AGE
hello-secure-6475f657c9-rmgsr   0/1     CrashLoopBackOff   4          2m45s
```

- 3.3. Examinez les journaux du pod qui a échoué. Les journaux indiquent que le pod tente d'utiliser le fichier `/run/secrets/nginx/tls.crt`, mais que ce fichier n'existe pas.

```
[student@workstation review-template]$ oc logs hello-secure-6475f657c9-rmgsr
...output omitted...
nginx: [emerg] BIO_new_file("/run/secrets/nginx/tls.crt") failed (SSL:
error:02001002:system library:fopen:No such file or directory:fopen('/run/
secrets/nginx/tls.crt','r') error:2006D080:BIO routines:BIO_new_file:no such file)
```

- 3.4. Utilisez la commande `oc set volumes` pour monter le secret dans le répertoire `/run/secrets/nginx`.

```
[student@workstation review-template]$ oc set volumes deployment/hello-secure \
>   --add --type secret --secret-name hello-tls --mount-path /run/secrets/nginx
info: Generated volume name: volume-hlrf
deployment.apps/hello-secure volume updated
```

- 3.5. Vérifiez que le pod `hello-secure` est redéployé correctement.

```
[student@workstation review-template]$ watch oc get pods
```

Appuyez sur `Ctrl+C` pour mettre fin à la commande `watch` une fois que le pod affiche `1/1` et `Running`.

```
Every 2.0s: oc get pods
...
NAME                  READY   STATUS        RESTARTS   AGE
hello-secure-6bd8fcccb4-nhwr2   1/1     Running      0          25s
```

4. Le certificat `hello-secure-combined.pem` est valide pour un seul nom d'hôte. Utilisez la commande `openssl x509` avec les options `-noout` et `-ext 'subjectAltName'` pour lire le certificat `hello-secure-combined.pem` et identifier le nom d'hôte. En tant qu'utilisateur `developer`, créez une route directe vers le service `hello-secure` à l'aide du nom d'hôte identifié. Vérifiez que la route répond aux requêtes externes.

**Note**

La page man `x509(1)` fournit des informations sur la commande `openssl x509`.

- 4.1. Examinez le certificat `hello-secure-combined.pem` à l'aide de la commande `openssl x509`.

```
[student@workstation review-template]$ openssl x509 \
> -in hello-secure-combined.pem -noout -ext 'subjectAltName'
X509v3 Subject Alternative Name:
DNS:hello-secure.apps.ocp4.example.com
```

- 4.2. Créez une route directe vers le service `hello-secure` pointant vers `hello-secure.apps.ocp4.example.com`.

```
[student@workstation review-template]$ oc create route passthrough \
> --service hello-secure --hostname hello-secure.apps.ocp4.example.com
route.route.openshift.io/hello-secure created
```

- 4.3. Revenez au répertoire `/home/student`.

```
[student@workstation review-template]$ cd
```

Utilisez la commande `curl` pour vérifier que la route répond aux requêtes externes.

```
[student@workstation ~]$ curl -s https://hello-secure.apps.ocp4.example.com \
> | grep -i Hello
<h1>Hello, world from nginx!</h1>
```

5. En tant qu'utilisateur `developer`, configurez le déploiement `hello-secure` pour qu'il soit mis à l'échelle automatiquement. Il doit y avoir au moins un pod en cours d'exécution dans le déploiement. Si l'utilisation moyenne du processeur est supérieure à 80 %, le déploiement augmente sur un maximum de cinq pods.

**Note**

Vous pouvez utiliser le script situé dans `~/D0280/solutions/review-template/test-hpa.sh` pour vérifier que votre déploiement est mis à l'échelle comme prévu.

- 5.1. Utilisez la commande `oc autoscale` pour créer la ressource de mise à l'échelle horizontale automatique du pod.

```
[student@workstation ~]$ oc autoscale deployment/hello-secure \
> --min 1 --max 5 --cpu-percent 80
horizontalpodautoscaler.autoscaling/hello-secure autoscaled
```

- 5.2. Exécutez le script `test-hpa.sh` fourni. Le script utilise la commande `ab` pour appliquer la charge à l'aide de l'outil d'étalonnage Apache.

```
[student@workstation ~]$ ~/DO280/solutions/review-template/test-hpa.sh  
...output omitted...  
Benchmarking hello-secure.apps.ocp4.example.com (be patient)  
Completed 10000 requests  
Completed 20000 requests  
...output omitted...  
Finished 100000 requests  
...output omitted...
```

- 5.3. Vérifiez que le nombre de pods `hello-secure` en cours d'exécution est compris entre deux et cinq.

```
[student@workstation ~]$ oc get pods  
NAME          READY   STATUS    RESTARTS   AGE  
hello-secure-6bd8fcccbb4-7qjcz  1/1     Running   0          96s  
hello-secure-6bd8fcccbb4-d67xd  1/1     Running   0          66s  
hello-secure-6bd8fcccbb4-m8vxp  1/1     Running   0          96s  
hello-secure-6bd8fcccbb4-nhwr2  1/1     Running   0          9m36s
```

## Évaluation

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour évaluer votre travail. Corrigez toute erreur signalée et répétez la commande tant que des erreurs persistent.

```
[student@workstation ~]$ lab review-template grade
```

## Fin

En tant qu'utilisateur `student` sur la machine `workstation`, utilisez la commande `lab` pour effectuer cet exercice.

```
[student@workstation ~]$ lab review-template finish
```

L'atelier est maintenant terminé.



## annexe A

# Création d'un compte GitHub

### Objectif

Décrire comment créer un compte GitHub pour les ateliers du cours.

# Création d'un compte GitHub

## Résultats

À la fin de cette section, vous devez pouvoir créer un compte GitHub et de créer des référentiels Git publics pour les ateliers de ce cours.

## Création d'un compte GitHub

Vous avez besoin d'un compte GitHub pour créer un ou plusieurs référentiels Git *public* pour les ateliers de ce cours. Si vous disposez déjà d'un compte GitHub, vous pouvez ignorer les étapes répertoriées dans cette annexe.



### Important

Si vous disposez déjà d'un compte GitHub, vérifiez que vous créez uniquement des référentiels Git *public* pour les ateliers de ce cours. Les scripts et instructions de notation de l'atelier nécessitent un accès non authentifié pour cloner le référentiel. Les référentiels doivent être accessibles sans avoir à fournir de mot de passe, de clé SSH ou de clé GPG.

Pour créer un nouveau compte GitHub, procédez comme suit :

1. Accédez à <https://github.com> à l'aide d'un navigateur Web.
2. Entrez les informations requises, puis cliquez **Create account**.

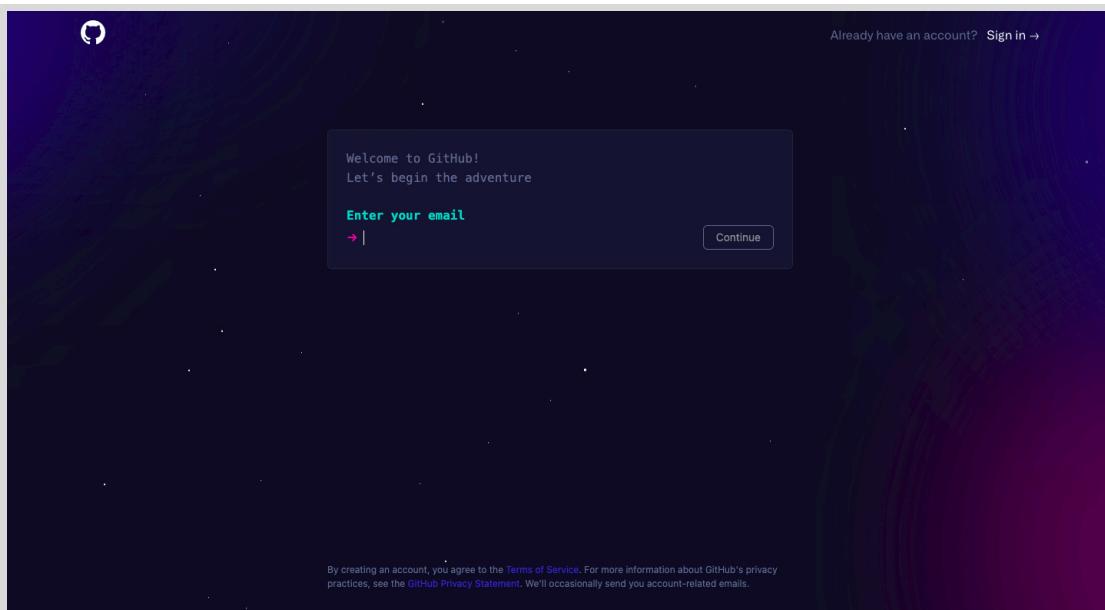
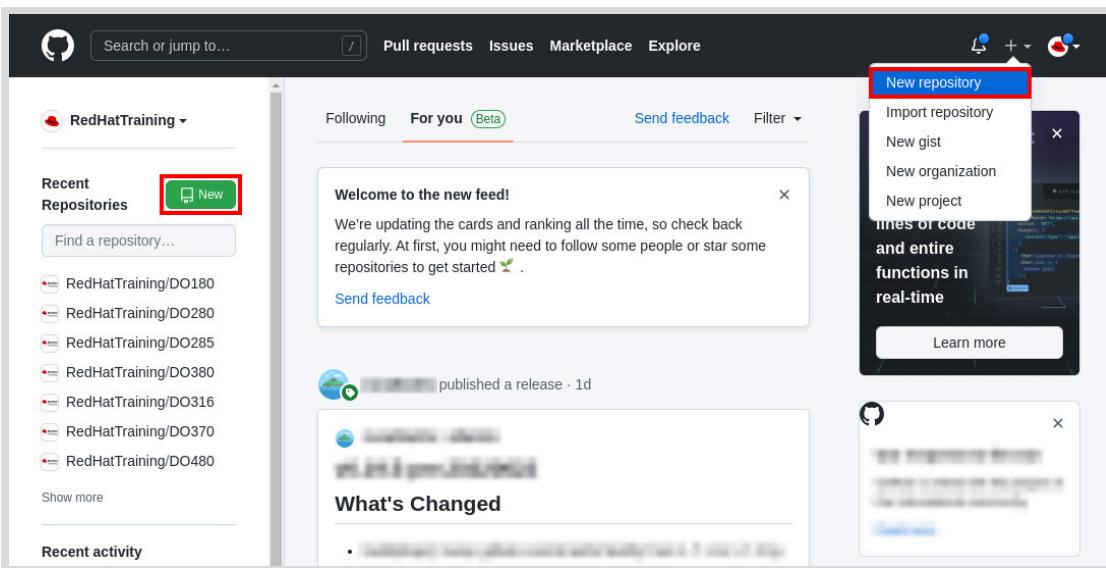


Figure A.1: Création d'un compte GitHub

3. Vous recevez un e-mail avec des instructions sur la façon d'activer votre compte GitHub. Vérifiez votre adresse électronique, puis connectez-vous au site Web GitHub à l'aide du nom d'utilisateur et du mot de passe que vous avez fournis lors de la création du compte.

4. Une fois que vous vous êtes connecté à GitHub, vous pouvez créer de nouveaux référentiels Git en cliquant sur **New** dans le volet **Repositories** à gauche de la page d'accueil de GitHub.



**Figure A.2: Création d'un nouveau référentiel Git**

Vous pouvez également cliquer sur l'icône plus (+) dans le coin supérieur droit (à droite de l'icône en forme de cloche), puis cliquer sur **New repository**.

## Références

**Signing up for a new GitHub account**  
<https://docs.github.com/en/get-started/signing-up-for-github/signing-up-for-a-new-github-account>

**Création d'un référentiel**  
<https://docs.github.com/en/repositories/creating-and-managing-repositories/creating-a-new-repository>



## annexe B

# Création d'un compte Quay

### Objectif

Décrire comment créer un compte Quay pour les ateliers du cours.

# Création d'un compte Quay

## Résultats

À la fin de cette section, vous serez en mesure de créer un compte Quay et de créer des référentiels d'images de conteneur publics pour les exercices pratiques de ce cours.

## Création d'un compte Quay

Vous avez besoin d'un compte Quay pour créer un ou plusieurs référentiels d'images de conteneur *public* pour les exercices pratiques de ce cours. Si vous disposez déjà d'un compte Quay, vous pouvez ignorer les étapes de création d'un nouveau compte répertorié dans cette annexe.



### Important

Si vous disposez déjà d'un compte Quay, vérifiez que vous créez uniquement des référentiels d'images de conteneur *public* pour les exercices pratiques de ce cours. Les scripts et instructions de notation de l'atelier nécessitent un accès non authentifié pour extraire des images de conteneur à partir du référentiel.

Pour créer un nouveau compte Quay, procédez comme suit :

1. Accédez à <https://quay.io> à l'aide d'un navigateur Web.
2. Cliquez sur **Sign in** dans le coin supérieur droit (en regard de la barre de recherche).
3. Sur la page **Sign in**, vous pouvez vous connecter à l'aide de vos informations d'identification Red Hat (crées dans l'Annexe C, *Création d'un compte Red Hat*).

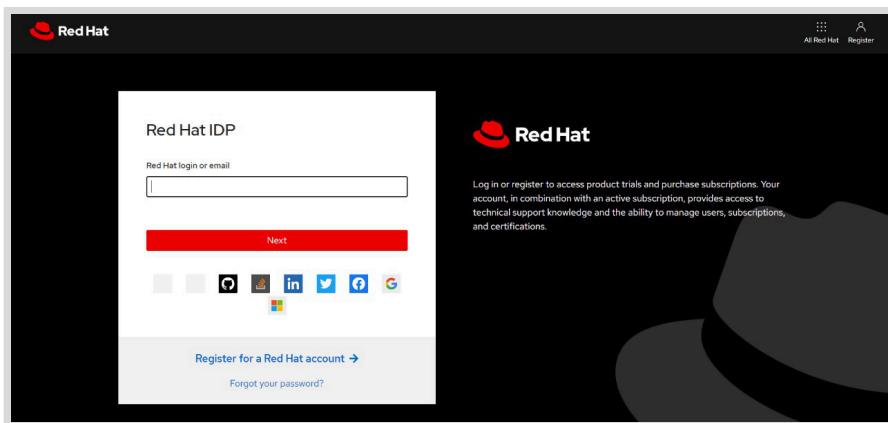


Figure B.1: Connectez-vous à l'aide de vos informations d'identification Red Hat.

Après vous être connecté à Quay, vous pouvez créer de nouveaux référentiels d'images en cliquant sur **Create New Repository** dans la page **Repositories**.

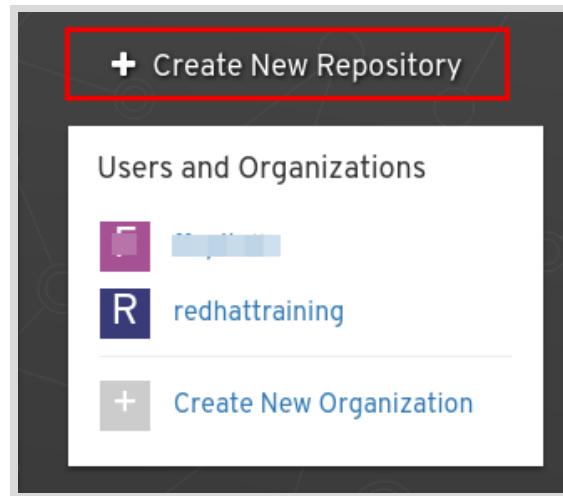


Figure B.2: Création d'un référentiel d'images

Vous pouvez également cliquer sur l'icône « plus » (+) dans le coin supérieur droit (à gauche de l'icône en forme de cloche), puis cliquer sur **New Repository**.

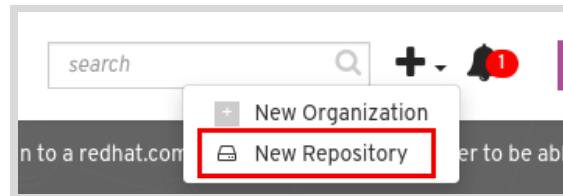


Figure B.3: Création d'un référentiel d'images

## Utilisation des outils de l'interface de ligne de commande

Si vous avez créé votre compte avec Red Hat, Google ou Github, vous devez définir un mot de passe de compte pour utiliser des outils d'interface de ligne de commande tels que Podman ou Docker.

1. Cliquez sur <YOUR\_USERNAME> en haut à droite.
2. Cliquez sur **Account Settings**.
3. Cliquez sur **User Settings** dans la colonne de gauche.
4. Cliquez sur le lien **Set password**.



### Références

#### Prise en main de Quay.io

<https://docs.quay.io/solution/getting-started.html>

# Visibilité des référentiels

## Résultats

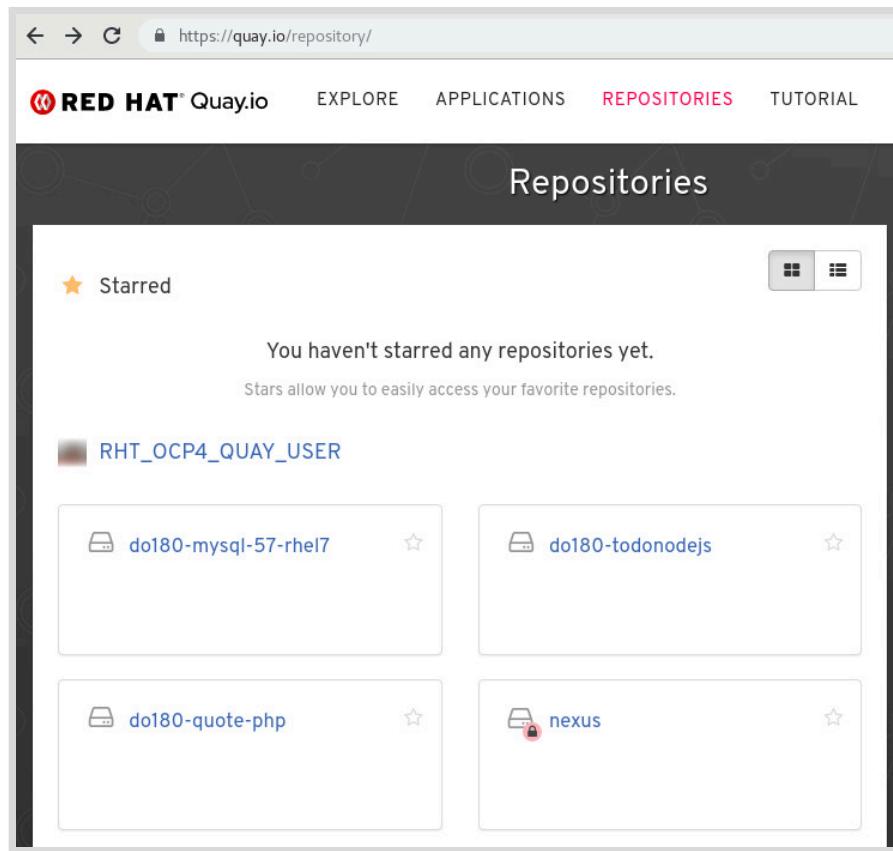
À la fin de cette section, vous devez pouvoir contrôler la visibilité des référentiels sur Quay.io.

### Visibilité du référentiel Quay.io

Quay.io offre la possibilité de créer des référentiels publics et privés. Les référentiels publics peuvent être lus par tout le monde sans aucune restriction, même si les autorisations en écriture doivent être accordées explicitement. Dans le cas des référentiels privés, les autorisations en lecture et en écriture sont limitées. Cependant, le nombre de référentiels privés dans quay.io est limité, en fonction du plan de l'espace de noms.

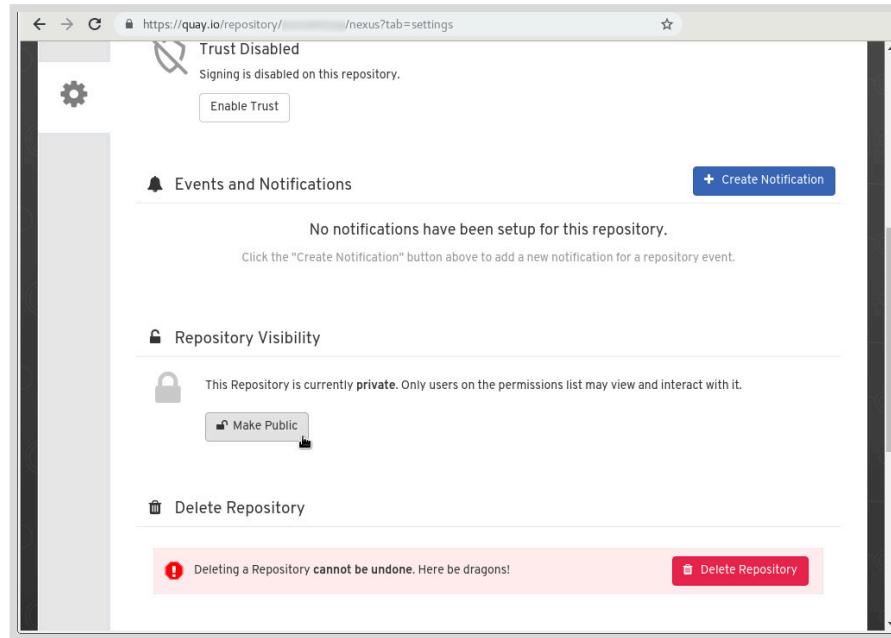
### Visibilité des référentiels par défaut

Les référentiels créés en envoyant par push des images à quay.io sont privés par défaut. Pour qu'OpenShift (ou tout autre outil) puisse récupérer ces images, vous pouvez soit configurer une clé privée dans OpenShift et dans Quay, soit rendre le référentiel public, de sorte qu'aucune authentification ne soit requise. La configuration de clés privées sort du cadre de ce document.



## Mise à jour de la visibilité des référentiels

Pour définir la visibilité des référentiels sur « publique », sélectionnez le référentiel approprié à l'adresse <https://quay.io/repository/> (connectez-vous à votre compte si nécessaire) et ouvrez la page **Settings** en cliquant sur l'icône d'engrenage en bas à gauche. Faites défiler vers le bas jusqu'à la section **Repository Visibility**, puis cliquez sur le bouton **Make public**.



Revenez à la liste des référentiels. L'icône du verrou en regard du nom du référentiel a disparu, ce qui indique que le référentiel est devenu public.



## annexe C

# Création d'un compte Red Hat

### Objectif

Décrire comment créer un compte Red Hat pour les ateliers du cours.

# Création d'un compte Red Hat

## Résultats

À la fin de cette section, vous devez pouvoir créer un compte Red Hat pour accéder aux images Red Hat Container Catalog pour les ateliers de ce cours.

### Création d'un compte Red Hat

Vous avez besoin d'un compte Red Hat pour accéder aux images Red Hat Container Catalog. Si vous disposez déjà d'un compte Red Hat, vous pouvez ignorer ces étapes.

Pour créer un nouveau compte Red Hat, procédez comme suit :

1. Accédez à <https://redhat.com> à l'aide d'un navigateur Web.
2. Cliquez sur **Log in** en haut à droite.
3. Cliquez sur **Register now** dans le volet de droite.

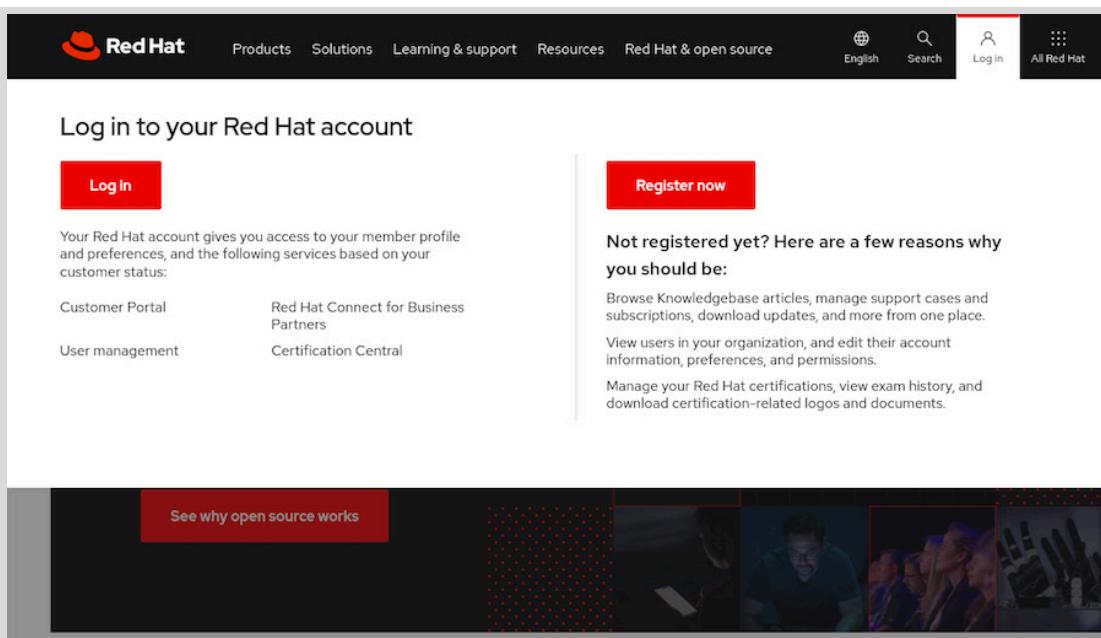
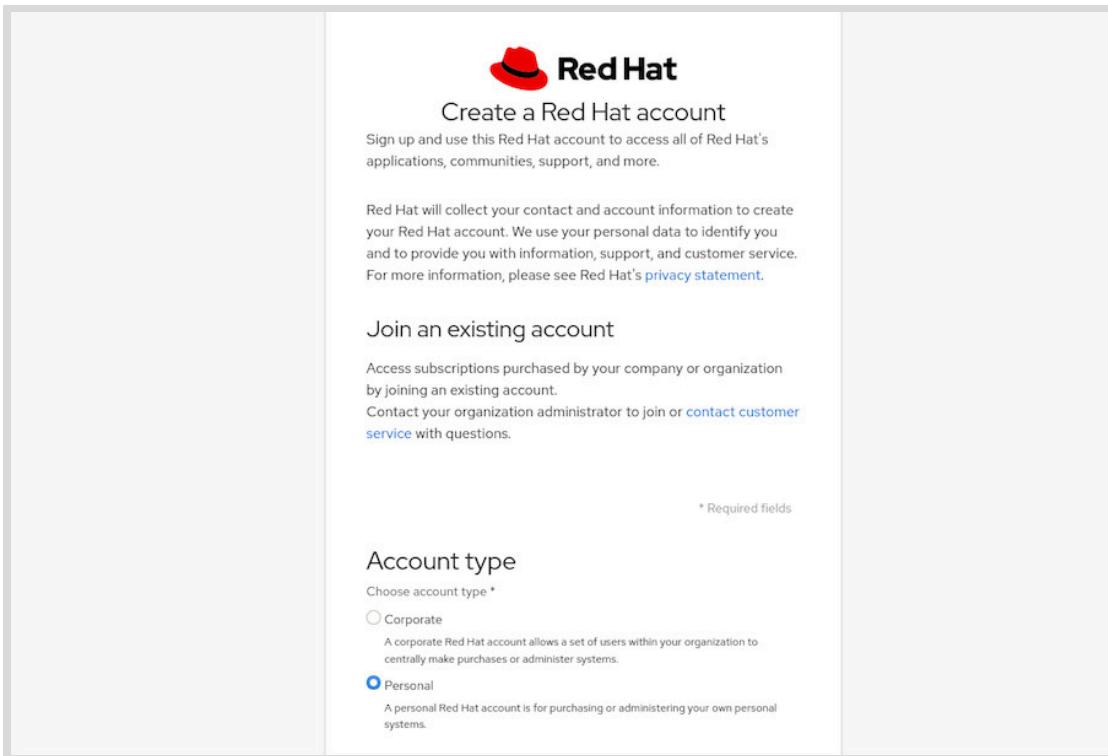


Figure C.1: Enregistrement d'un nouveau compte

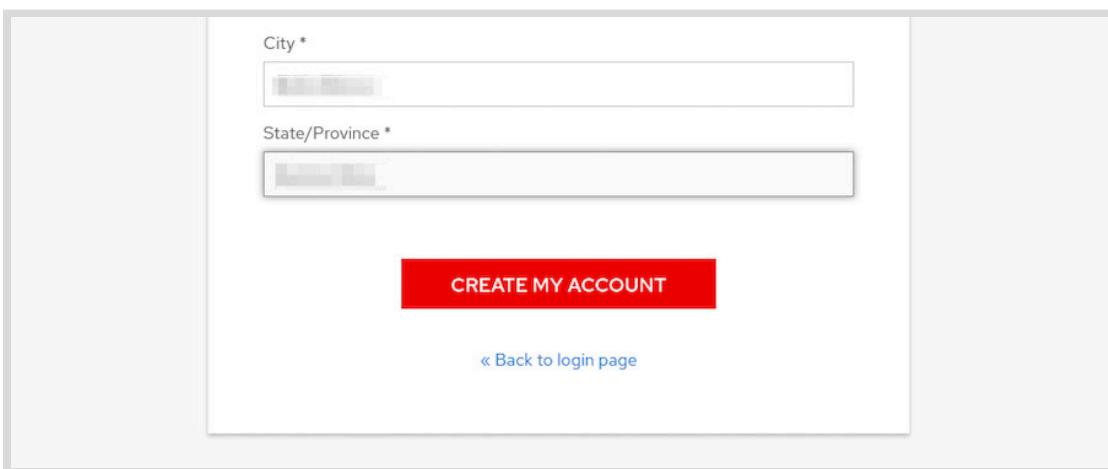
4. Définissez Account type sur Personal.



The screenshot shows the 'Create a Red Hat account' page. At the top is the Red Hat logo. Below it is the heading 'Create a Red Hat account'. A sub-instruction says 'Sign up and use this Red Hat account to access all of Red Hat's applications, communities, support, and more.' A note below states: 'Red Hat will collect your contact and account information to create your Red Hat account. We use your personal data to identify you and to provide you with information, support, and customer service. For more information, please see Red Hat's [privacy statement](#)'. A section titled 'Join an existing account' explains how to access subscriptions by joining an existing account or contacting customer service. A note at the bottom right indicates '\* Required fields'.

**Figure C.2: Définition d'un type de compte**

5. Renseignez le reste du formulaire avec vos informations personnelles. Dans ce formulaire, vous sélectionnez également le nom d'utilisateur et le mot de passe de ce compte.
6. Cliquez sur CREATE MY ACCOUNT.



The screenshot shows a 'Personal' account type selection screen. It includes fields for 'City \*' (with a placeholder redacted), 'State/Province \*' (with a placeholder redacted), and a large red 'CREATE MY ACCOUNT' button. Below the button is a link '« Back to login page'.

**Figure C.3: Renseigner le formulaire d'informations personnelles**



## Références

### Portail client Red Hat

<https://access.redhat.com/>



## annexe D

# Commandes Git utiles

### Objectif

Décrire les commandes Git utiles qui sont utilisées pour les exercices pratiques de ce cours.

# Commandes Git

---

## Résultats

À la fin de cette section, vous serez en mesure de redémarrer et de refaire des exercices dans ce cours. Vous devez également être en mesure de passer d'un exercice incomplet pour en effectuer un autre, puis de reprendre l'exercice précédent là où vous vous êtes arrêté.

## Utilisation des branches Git

Ce cours utilise un référentiel Git hébergé sur GitHub pour stocker le code source du cours de l'application. Au début du cours, vous créez votre propre branche de ce référentiel, qui est également hébergé sur GitHub.

Au cours de ce cours, vous utilisez une copie locale de votre branche, que vous clonez sur la machine virtuelle `workstation`. Le terme `origin` fait référence au référentiel distant à partir duquel un référentiel local est cloné.

Au fur et à mesure que vous parcourez les exercices du cours, vous utilisez des branches Git séparées pour chaque exercice. Toutes les modifications que vous apportez au code source se produisent dans une nouvelle branche que vous créez uniquement pour les besoins de cet exercice. N'apportez jamais aucune modification à la branche `master`.

La liste ci-dessous répertorie les scénarios et les commandes Git correspondantes que vous pouvez utiliser pour manipuler des branches, et pour revenir à un état correct connu.

## Recommencer un exercice à partir de zéro

Pour recommencer un exercice à partir de zéro une fois que vous l'avez terminé, procédez comme suit :

- Vous validez et transmettez toutes les modifications de votre branche locale dans le cadre de la réalisation de l'exercice. Vous avez terminé l'exercice en exécutant sa sous-commande `finish` pour nettoyer toutes les ressources :

```
[student@workstation ~]$ lab your-exercise finish
...output omitted...
```

- Changez à votre clon local du référentiel `D0180-apps` et basculez sur la branche `master`:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- Supprimez votre branche locale :

```
[student@workstation D0180-apps]$ git branch -d your-branch
...output omitted...
```

**annexe D |** Commandes Git utiles

4. Supprimez la branche distante sur votre compte GitHub personnel :

```
[student@workstation D0180-apps]$ git push origin --delete your-branch  
...output omitted...
```

5. Utilisez la sous-commande **start** pour redémarrer l'exercice :

```
[student@workstation D0180-apps]$ cd ~  
  
[student@workstation ~]$ lab your-exercise start  
...output omitted...
```

## Abandon d'un exercice partiellement terminé et redémarrage à partir de zéro

Vous pouvez être dans un scénario où vous avez partiellement effectué quelques étapes de l'exercice, et vous souhaitez abandonner la tentative en cours et la redémarrer à partir de zéro. Effectuez les étapes suivantes :

1. Exécutez la sous-commande **finish** de l'exercice pour nettoyer toutes les ressources.

```
[student@workstation ~]$ lab your-exercise finish  
...output omitted...
```

2. Entrez votre clone local du référentiel D0180-apps et ignorez toutes les modifications en attente sur la branche actuelle à l'aide de **git stash** :

```
[student@workstation ~]$ cd ~/D0180-apps  
  
[student@workstation D0180-apps]$ git stash  
...output omitted...
```

3. Basculez vers la branche **master** de votre référentiel local :

```
[student@workstation D0180-apps]$ git checkout master  
...output omitted...
```

4. Supprimez votre branche locale :

```
[student@workstation D0180-apps]$ git branch -d your-branch  
...output omitted...
```

5. Supprimez la branche distante sur votre compte GitHub personnel :

```
[student@workstation D0180-apps]$ git push origin --delete your-branch  
...output omitted...
```

6. Vous pouvez maintenant redémarrer l'exercice en exécutant sa sous-commande **start** :

```
[student@workstation D0180-apps]$ cd ~

[student@workstation ~]$ lab your-exercise start
...output omitted...
```

## Passage à un autre exercice à partir d'un exercice incomplet

Vous pouvez être dans un scénario dans lequel vous avez partiellement effectué quelques étapes d'un exercice, mais vous souhaitez passer à un autre exercice et revenir à l'exercice en cours ultérieurement.

Évitez de laisser un trop grand nombre d'exercices inachevés auxquels vous reviendrez plus tard. Ces exercices encombrent les ressources du cloud et il se peut que vous utilisez la totalité de votre quota alloué sur le fournisseur de cloud et sur le cluster OpenShift que vous partagez avec un autre stagiaire. Si vous pensez que vous ne reviendrez pas à l'exercice actuel avant un certain moment, envisagez de l'abandonner complètement pour le recommencer plus tard depuis le début.

Si vous préférez interrompre l'exercice en cours et travailler sur le suivant, procédez comme suit :

1. Validez toutes les modifications en attente dans votre référentiel local et transmettez-les à votre compte GitHub personnel. Vous souhaiterez peut-être enregistrer l'étape à laquelle vous avez arrêté l'exercice :

```
[student@workstation ~]$ cd ~/D0180-apps

[student@workstation D0180-apps]$ git commit -a -m "Paused at step X.Y"
...output omitted...
[student@workstation D0180-apps]$ git push
...output omitted...
```

2. N'exécutez pas la commande `finish` de l'exercice d'origine. Il est important de ne pas modifier vos projets OpenShift existants, de sorte que vous puissiez les reprendre ultérieurement.
3. Démarrez l'exercice suivant en exécutant sa sous-commande `start` :

```
[student@workstation ~]$ lab your-exercise start
...output omitted...
```

4. L'exercice suivant bascule vers la branche `master` et crée éventuellement une nouvelle branche pour ses modifications. Cela signifie que les modifications apportées à l'exercice d'origine dans la branche d'origine restent inchangées.
5. Ensuite, une fois que vous avez terminé l'exercice suivant et que vous souhaitez revenir à l'exercice d'origine, revenez à sa branche :

```
[student@workstation ~]$ git checkout original-branch
...output omitted...
```

Vous pouvez ensuite poursuivre l'exercice d'origine et revenir à l'étape où vous vous êtes arrêté.



## Références

### **Git branch man page**

<https://git-scm.com/docs/git-branch>

### **What is a Git branch?**

<https://git-scm.com/book/en/v2/Git-Branching-Banches-in-a-Nutshell>

### **Git Tools - Stashing**

<https://git-scm.com/book/en/v2/Git-Tools-Stashing-and-Cleaning>

