





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



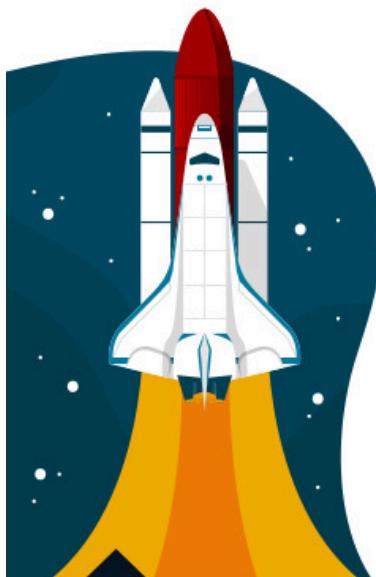
Engage in thousands of active conversations and posts



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



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.

Red Hat OpenShift

I : conteneurs & Kubernetes



OCP 4.10 DO180

Red Hat OpenShift I : conteneurs & Kubernetes

Édition 1 20220323

Date de publication 20220323

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

Copyright © 2021 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are
Copyright © 2021 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 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 et Fiona Allen

Conventions de la documentation	ix
Introduction	xi
DO180 : Red Hat OpenShift I : Containers & Kubernetes	xi
Organisation de l'environnement de la classe	xii
Internationalisation	xvi
1. Présentation de la technologie de conteneur	1
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	29
2. Création de services en conteneur	31
Approvisionnement de services en conteneur	32
Exercice guidé: Crédit d'une instance de base de données MySQL	38
Utilisation de conteneurs sans racine	41
Exercice guidé: Découverte des conteneurs avec et sans racine	44
Open Lab: Création de services en conteneur	46
Résumé	50
3. Gestion des conteneurs	51
Gestion du cycle de vie des conteneurs	52
Exercice guidé: Gestion d'un conteneur MySQL	60
Association du stockage persistant à des conteneurs	64
Exercice guidé: Créer un conteneur MySQL avec une base de données persistante	68
Accès aux conteneurs	71
Exercice guidé: Chargement de la base de données	73
Open Lab: Gestion des conteneurs	76
Résumé	85
4. Gestion des images de conteneur	87
Accès aux registres	88
Quiz: Utilisation des registres	95
Manipulation d'images de conteneur	99
Exercice guidé: Crédit d'une image de conteneur Apache personnalisée	105
Open Lab: Gestion des images	110
Résumé	118
5. Création d'images de conteneur personnalisées	119
Conception d'images de conteneur personnalisées	120
Quiz: Méthodes relatives à la conception d'images de conteneur	124
Compilation d'images de conteneur personnalisées avec des Containerfiles	126
Exercice guidé: Crédit d'une image de conteneur Apache de base	132
Open Lab: Crédit d'images de conteneur personnalisées	136
Résumé	143
6. Déploiement d'applications en conteneur dans OpenShift	145
Description de l'architecture de Kubernetes et d'OpenShift	146
Quiz: Description de Kubernetes et d'OpenShift	153
Création de ressources Kubernetes	157
Exercice guidé: Déploiement d'un serveur de base de données dans OpenShift	170
Création de routes	175
Exercice guidé: Exposition d'un service en tant que route	179

Création d'applications avec Source-to-Image	184
Exercice guidé: Création d'une application en conteneur avec Source-to-Image	194
Création d'applications avec la console Web OpenShift	200
Exercice guidé: Création d'une application à l'aide de la console Web	206
Open Lab: Déploiement d'applications en conteneur dans OpenShift	222
Résumé	226
7. Déploiement d'applications multiconteneurs	227
Considérations liées aux applications multiconteneurs	228
Exercice guidé: Déploiement de l'application Web et de conteneurs MySQL sur Linux	233
Déploiement d'une application multiconteneur sur OpenShift	238
Exercice guidé: Création d'une application sur OpenShift	240
Déploiement d'une application multiconteneur sur OpenShift à l'aide d'un modèle	244
Exercice guidé: Création d'une application avec un modèle	251
Open Lab: Déploiement d'applications multiconteneurs	255
Résumé	262
8. Résolution des problèmes relatifs aux applications en conteneur	263
Résolution des problèmes relatifs aux déploiements et aux builds S2I	264
Exercice guidé: Résolution des problèmes relatifs à une compilation OpenShift	270
Résolution des problèmes relatifs aux applications en conteneur	278
Exercice guidé: Configuration des journaux de conteneur Apache pour le débogage	285
Open Lab: Résolution des problèmes relatifs aux applications en conteneur	288
Résumé	299
9. Révision complète	301
Révision complète	302
Open Lab: Mise en conteneur et déploiement d'une application logicielle	304
A. Implémentation de l'architecture des microservices	315
Implémentation des architectures des microservices	316
Exercice guidé: Refactorisation de l'application To Do List	321
Résumé	326
B. Crédit d'un compte GitHub	327
Création d'un compte GitHub	328
C. Crédit d'un compte Quay	331
Création d'un compte Quay	332
Visibilité des référentiels	334
D. Crédit d'un compte Red Hat	337
Création d'un compte Red Hat	338
E. Commandes Git utiles	341
Commandes Git	342

Conventions de la documentation



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 considérée. Le fait d'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

Les cadres « Important » 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 un cadre « Important » 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.

Introduction

DO180 : Red Hat OpenShift I : Containers & Kubernetes

DO180 : Red Hat OpenShift I : Containers & Kubernetes est un cours pratique qui enseigne aux étudiants comment créer, déployer et gérer des conteneurs à l'aide de Podman, Kubernetes et de la plateforme Red Hat OpenShift Container Platform.

L'un des principaux tenants du mouvement DevOps est l'intégration continue et le déploiement continu. Les conteneurs sont devenus une technologie clé pour la configuration et le déploiement d'applications et de microservices. La plateforme Red Hat OpenShift Container Platform est une implémentation de Kubernetes, un système d'orchestration de conteneur.

Objectifs du cours

- Montrer la connaissance de l'écosystème des conteneurs.
- Gérer les conteneurs Linux avec Podman.
- Déployer des conteneurs sur un cluster Kubernetes à l'aide de la plateforme OpenShift Container Platform.
- Montrer la conception de base du conteneur et la capacité de construire des images de conteneur.
- Mettre en œuvre une architecture basée sur le conteneur en utilisant les connaissances des conteneurs, Kubernetes et OpenShift.

Public

- Administrateurs système
- Développeurs
- Responsables informatiques et architectes d'infrastructure

Conditions préalables

Les stagiaires doivent remplir une ou plusieurs des conditions préalables suivantes :

- Pouvoir utiliser une session de terminal Linux et émettre des commandes du système d'exploitation. Un certificat RHCSA valide est recommandé, mais pas obligatoire.
- Connaître les architectures d'applications Web et les technologies associées.

Organisation de l'environnement de la classe

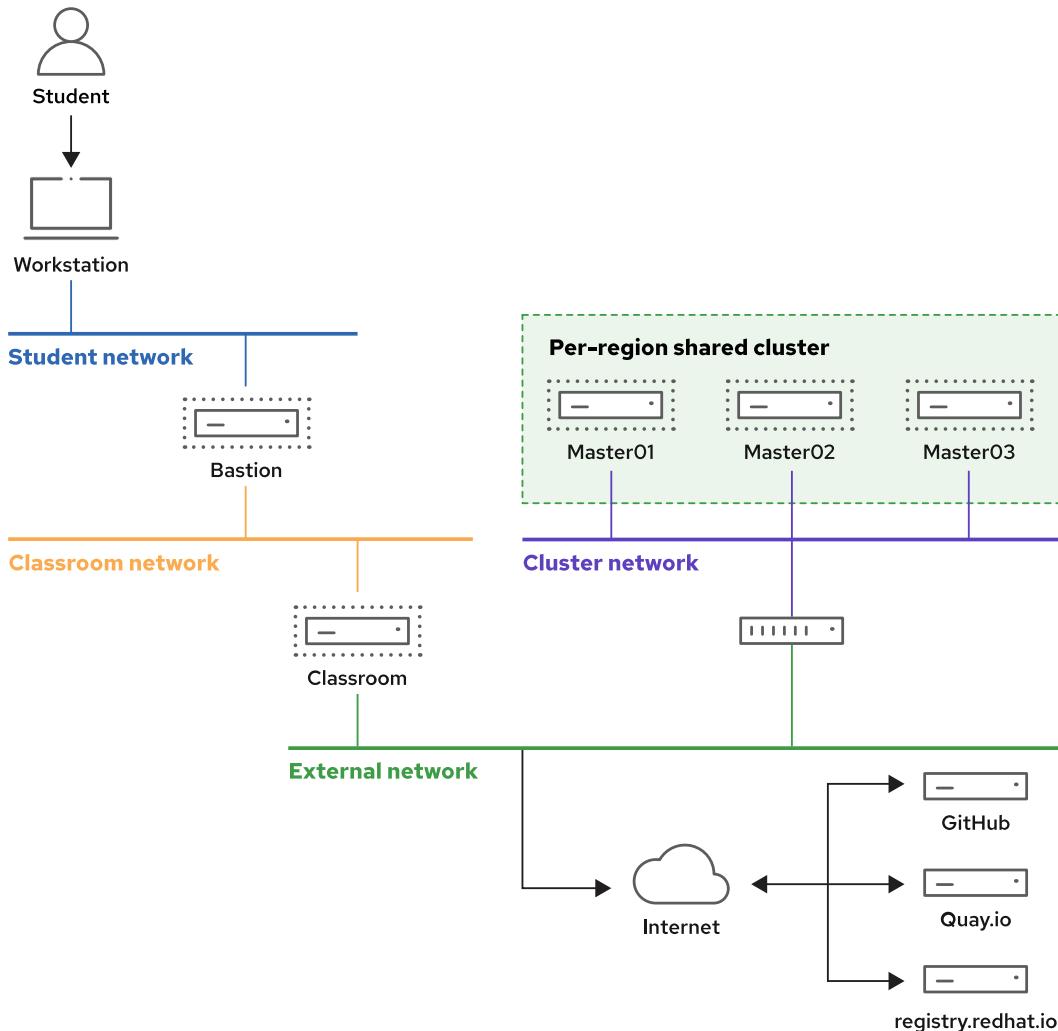


Figure 0.1: Environnement de formation

Dans ce cours, le système informatique principal utilisé pour les travaux pratiques est `workstation`. Il s'agit d'une machine virtuelle (VM) nommée `workstation.lab.example.com`.

Tous les systèmes informatiques des stagiaires possèdent un compte d'utilisateur standard, `student`, protégé par le mot de passe `student`. Le mot de passe `root` est `redhat` sur tous les systèmes des stagiaires.

Machines de la salle de classe

Nom de la machine	Adresses IP	Rôle
workstation.lab.example.com	172.25.250.9	Poste de travail graphique utilisé pour l'administration du système
classroom.example.com	172.25.254.254	Routeur reliant le réseau de la salle de classe à Internet
bastion.lab.example.com	172.25.252.1	Routeur reliant le réseau du stagiaire à celui de la salle de classe

Plusieurs systèmes dans la salle de classe proposent des services d'assistance. Deux serveurs, `content.example.com` et `materials.example.com`, hébergent les logiciels et les supports d'atelier utilisés pour les activités pratiques. Les informations relatives à l'utilisation de ces serveurs sont fournies dans les instructions de ces activités.

Les stagiaires utilisent la machine `workstation` pour accéder à un cluster OpenShift partagé hébergé en externe. Les stagiaires n'ont pas de priviléges d'administrateur de cluster sur le cluster, mais ce n'est pas nécessaire pour compléter le contenu DO180.

Les stagiaires ont à leur disposition un compte sur un cluster OpenShift 4 partagé lorsqu'ils approvisionnent leur environnement dans l'interface de formation en ligne de Red Hat. Des informations de cluster telles que le point d'accès de l'API et l'ID de cluster, ainsi que leur nom d'utilisateur et mot de passe leur sont présentés lorsqu'ils approvisionnent leur environnement.

Les stagiaires ont également accès à un MySQL et un serveur Nexus hébergés par le cluster OpenShift ou un fournisseur externe. Les activités pratiques de ce cours fournissent des instructions pour accéder à ces serveurs au besoin.

Les activités pratiques du cours DO180 exigent également que les stagiaires disposent des comptes personnels sur deux services Internet publics et gratuits : GitHub et Quay.io. Les stagiaires doivent créer ces comptes s'ils ne les ont pas déjà (voir l'annexe) et vérifier leur accès en se connectant à ces services avant de commencer la classe.

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 : `rol.redhat.com` [<http://rol.redhat.com>]. Pour se connecter à ce site, les stagiaires doivent utiliser leurs informations d'identification du Portail client Red Hat.

Contrôle des machines virtuelles

Les machines virtuelles de votre environnement de formation sont contrôlées sur une Page Web. L'état de chaque machine virtuelle de la salle de classe est affiché sur la page de l'onglet Lab Environment.

États de la machine

État de la machine virtuelle	Description
active	La machine virtuelle est en cours d'exécution et disponible (ou, pendant le démarrage, le sera bientôt.)

État de la machine virtuelle	Description
stopped	La machine virtuelle est complètement arrêtée.
building	La création initiale de la machine virtuelle est en cours.

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

Actions de machine/salle de classe

Bouton ou action	Description
CREATE	Permet de créer la salle de classe ROL. Crée toutes les machines virtuelles nécessaires pour la salle de classe et les démarre. Cela peut prendre plusieurs minutes.
DELETE	Permet de supprimer la salle de classe ROL. Détruit toutes les machines virtuelles dans la salle de classe. Attention : tous les travaux générés sur les disques seront perdus.
START	Permet de démarrer toutes les machines virtuelles de la salle de classe.
OPEN CONSOLE	Permet d'ouvrir un nouvel onglet dans le navigateur et de se connecter à la console de la machine virtuelle. Les stagiaires peuvent se connecter directement à la machine virtuelle et exécuter des commandes. Dans la plupart des cas, les stagiaires doivent se connecter à la machine virtuelle workstation et utiliser ssh pour se connecter aux autres machines virtuelles.
Démarrer	Permet de démarrer (allumer) la machine virtuelle.
Arrêter	Permet d'éteindre correctement la machine virtuelle, en conservant le contenu sur son disque.
Mettre hors tension	Force l'arrêt de la machine virtuelle, en conservant le contenu du disque. Cela équivaut à couper l'alimentation d'une machine physique.
Réinitialiser	Force l'arrêt de la machine virtuelle et réinitialise le disque à son état initial. Attention : tout travail généré sur le disque va être perdu.

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, cliquez 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 aurez terminés jusqu'ici dans l'environnement de formation seront perdus.

Minuterie d'arrêt automatique

L'inscription à la formation en ligne Red Hat (ROL) confère aux stagiaires le droit d'utiliser l'ordinateur pendant un temps donné. Afin de les aider à conserver le temps d'utilisation de l'ordinateur qui leur est alloué, une minuterie d'arrêt automatique est associée à la salle de classe ROL. Celle-ci ferme l'environnement de formation à l'expiration du temps prévu.

Pour ajuster le timer, cliquez sur [+] pour ajouter une heure. Notez qu'il existe une durée maximale de douze heures.

Internationalisation

Sélection de la langue par utilisateur

Il se peut que vos utilisateurs veuillent utiliser, pour leur environnement de bureau, une langue différente de celle utilisée par l'ensemble du système. Il se peut aussi qu'ils veuillent utiliser une autre disposition de clavier ou une autre méthode de saisie pour leur compte.

Paramètres linguistiques

Dans l'environnement de travail GNOME, l'utilisateur peut être invité, lors de sa première connexion, à configurer la langue et la méthode de son choix. Dans le cas contraire, la manière la plus simple pour un utilisateur d'ajuster les réglages de langue et de méthode de saisie est d'utiliser l'application **Region & Language**.

Vous pouvez démarrer cette application de deux manières. Vous pouvez exécuter la commande `gnome-control-center region` depuis la fenêtre de terminal ou sur la barre du haut, à partir du menu système situé dans le coin droit, sélectionnez le bouton des paramètres (dont l'icône ressemble à un tournevis croisé et une clé) en bas à gauche du menu.

Dans la fenêtre qui s'ouvre, sélectionnez **Region & Language**. Cliquez sur la case **Language** et sélectionnez la langue souhaitée dans la liste qui s'affiche. Cela met également à jour le réglage **Formats** pour qu'il corresponde aux réglages par défaut pour cette langue. Ces modifications entreront en vigueur la prochaine fois que vous vous connectez.

Ces paramètres affectent l'environnement de bureau GNOME et toutes les applications qui y sont lancées, telles que `gnome-terminal`. Toutefois, par défaut, ils ne s'appliquent pas à ce compte si l'accès a été réalisé via une connexion ssh à partir d'un système distant ou d'une connexion texte sur une console virtuelle (telle que `tty5`).



Note

Vous pouvez faire en sorte que votre environnement de shell utilise le même paramètre `LANG` que votre environnement graphique, même lorsque vous vous connectez par l'intermédiaire d'une console virtuelle en mode texte ou par ssh. Pour ce faire, vous pouvez placer le code suivant ou son équivalent dans votre fichier `~/.bashrc`. Ce code fourni en exemple règle la langue utilisée pour une connexion en mode texte pour qu'elle corresponde à celle configurée pour l'environnement de bureau GNOME :

```
i=$(grep 'Language=' /var/lib/AccountsService/users/${USER} \
    | sed 's/Language=/"/')
if [ "$i" != "" ]; then
    export LANG=$i
fi
```

Le japonais, le coréen, le chinois et d'autres langues à jeu de caractères autre que le latin peuvent ne pas s'afficher correctement sur les consoles virtuelles en mode texte.

Introduction

On peut obliger chaque commande à utiliser une autre langue, en réglant la variable LANG depuis la ligne de commande :

```
[user@host ~]$ LANG=fr_FR.utf8 date  
jeu. avril 25 17:55:01 CET 2019
```

Les commandes suivantes continuent d'utiliser la langue par défaut du système. La commande locale peut être utilisée pour déterminer la valeur actuelle de LANG, ainsi que d'autres variables d'environnement connexes.

Paramètres de la méthode de saisie

Dans Red Hat Enterprise Linux 7 ou version ultérieure, GNOME 3 utilise automatiquement le système de sélection de méthode de saisie IBus qui permet de changer facilement et rapidement la disposition du clavier et les méthodes de saisie.

L'application Region & Language peut aussi servir à activer d'autres méthodes de saisie. Dans la fenêtre de l'application Region & Language, le cadre [Input Sources] présente les méthodes de saisie actuellement disponibles. Par défaut, [English (US)] peut être la seule méthode disponible. Sélectionnez [English (US)], puis cliquez sur l'icône du clavier pour afficher la disposition actuelle du clavier.

Pour ajouter une nouvelle méthode de saisie, cliquez sur le bouton [+] dans le coin inférieur gauche de la fenêtre [Input Sources]. Une fenêtre [Add an Input Source] s'ouvre. Sélectionnez votre langue, puis la méthode de saisie ou la disposition de clavier souhaitée.

Lorsque plusieurs méthodes de saisie ont été configurées. L'utilisateur peut passer rapidement de l'une à l'autre en saisissant [Super+Space] (parfois appelé [Windows+Space]). Par ailleurs, un indicateur d'état s'affiche dans la barre supérieure de l'environnement GNOME. Celui-ci a deux fonctions : il indique la méthode de saisie active et joue le rôle de menu vous permettant de passer d'une méthode de saisie à l'autre ou de sélectionner les fonctions avancées de méthodes de saisie plus complexes.

Certaines des méthodes sont marquées par des engrenages, qui indiquent qu'elles ont des options de configuration et des possibilités avancées. Par exemple, la méthode de saisie japonaise [Japonais (Kana Kanji)] permet à l'utilisateur de préparer un texte en caractères latins et d'utiliser les touches [Flèche vers le bas] et [Flèche vers le haut] pour sélectionner les caractères à utiliser.

Les anglophones américains peuvent également trouver cette méthode utile. Pour [English (United States)] par exemple, la disposition de clavier est [English (international avec touches mortes en AltGr)], qui considère la touche [AltGr] (ou la touche [Alt]) de droite) sur un clavier PC à 104-105 touches comme une touche de modification « Maj secondaire » et comme touche d'activation des touches mortes pour la saisie des caractères supplémentaires. Le Dvorak et d'autres dispositions sont également proposées.



Note

Si vous connaissez le point de code Unicode du caractère, vous pouvez le saisir dans l'environnement de bureau GNOME. Appuyez sur [Ctrl+Maj+U], suivi du point de code. Après avoir appuyé sur les touches [Ctrl+Maj+U], un u souligné s'affiche pour indiquer que le système attend la saisie du code du caractère.

Par exemple, la lettre minuscule grecque lambda a pour point de code U+03BB et peut être saisie en appuyant sur [Ctrl+Maj+U], puis 03BB, et ensuite [Entrée].

Paramètres linguistiques par défaut pour l'ensemble du système

La langue du système est réglée par défaut sur Anglais US. Elle fait appel à l'encodage Unicode UTF-8 (`en_US.utf8`) pour son jeu de caractères, mais celui-ci peut être modifié lors de l'installation ou plus tard.

Depuis la ligne de commande, l'utilisateur `root` peut modifier les paramètres linguistiques à l'échelle du système à l'aide de la commande `localectl`. Si la commande `localectl` est exécutée sans argument, elle affiche les paramètres linguistiques à l'échelle du système.

Pour définir une langue par défaut au niveau du système, exécutez la commande `localectl set-locale LANG=locale`, où `locale` est la valeur appropriée pour la variable d'environnement `LANG` correspondante décrite dans le tableau « Référence des codes de langue » du présent chapitre. Les modifications seront prises en compte lors de la prochaine connexion de l'utilisateur et seront enregistrées dans le fichier `/etc/locale.conf`.

```
[root@host ~]# localectl set-locale LANG=fr_FR.UTF8
```

Dans GNOME, les administrateurs peuvent modifier ce paramètre dans `Region & Language` en cliquant sur le bouton [Login Screen] dans le coin supérieur droit de la fenêtre. La modification de la langue [Language] de l'écran de connexion graphique ajustera également le paramètre qui définit la langue pour l'ensemble du système, stocké dans le fichier de configuration `/etc/locale.conf`.



Important

Les consoles virtuelles en mode texte, telles que `tty4`, sont plus limitées en ce qui concerne les polices qu'elles peuvent afficher que les terminaux d'une console virtuelle fonctionnant sous un environnement graphique, ou les pseudoterminalas pour les sessions `ssh`. Par exemple, les caractères japonais, coréens et chinois peuvent ne pas s'afficher correctement dans une console virtuelle en mode texte. Pour cette raison, vous devriez envisager d'utiliser l'anglais ou une autre langue avec un jeu de caractères latins pour la langue par défaut pour l'ensemble du système.

De même, les consoles virtuelles en mode texte reconnaissent moins de méthodes de saisie. Ce point est géré séparément de l'environnement graphique du bureau. On peut configurer les paramètres de saisie globaux par l'intermédiaire de `localectl`, à la fois pour les consoles virtuelles en mode texte et pour l'environnement graphique. Voir les pages du manuel `localectl(1)` et `vconsole.conf(5)` pour plus d'informations.

Modules linguistiques

Des paquetages RPM spéciaux appelés langpacks installent des paquetages de langue qui prennent en charge des langues spécifiques. Ces langpacks utilisent des dépendances pour installer automatiquement des paquetages RPM supplémentaires contenant des localisations, des dictionnaires et des traductions pour les autres paquetages logiciels de votre système.

Pour lister les langpacks installés et susceptibles d'être installés, utilisez `yum list langpacks-* :`

```
[root@host ~]# yum list langpacks-*  
Updating Subscription Management repositories.  
Updating Subscription Management repositories.  
Installed Packages  
langpacks-en.noarch      1.0-12.el8          @AppStream  
Available Packages  
langpacks-af.noarch       1.0-12.el8          rhel-8-for-x86_64-appstream-rpms  
langpacks-am.noarch       1.0-12.el8          rhel-8-for-x86_64-appstream-rpms  
langpacks-ar.noarch       1.0-12.el8          rhel-8-for-x86_64-appstream-rpms  
langpacks-as.noarch       1.0-12.el8          rhel-8-for-x86_64-appstream-rpms  
langpacks-ast.noarch      1.0-12.el8          rhel-8-for-x86_64-appstream-rpms  
...output omitted...
```

Pour ajouter une prise en charge linguistique, installez le paquetage langpacks approprié. Par exemple, la commande suivante ajoute la prise en charge du français :

```
[root@host ~]# yum install langpacks-fr
```

Utilisez `yum repoquery --what supplements` pour déterminer quels paquetages RPM peuvent être installés par un langpack :

```
[root@host ~]# yum repoquery --what supplements langpacks-fr  
Updating Subscription Management repositories.  
Updating Subscription Management repositories.  
Last metadata expiration check: 0:01:33 ago on Wed 06 Feb 2019 10:47:24 AM CST.  
glibc-langpack-fr-0:2.28-18.el8.x86_64  
gnome-getting-started-docs-fr-0:3.28.2-1.el8.noarch  
hunspell-fr-0:6.2-1.el8.noarch  
hyphen-fr-0:3.0-1.el8.noarch  
libreoffice-langpack-fr-1:6.0.6.1-9.el8.x86_64  
man-pages-fr-0:3.70-16.el8.noarch  
mythes-fr-0:2.3-10.el8.noarch
```



Important

Les paquetages langpacks utilisent les dépendances faibles RPM afin d'installer des paquetages supplémentaires uniquement lorsque le paquetage principal qui en a besoin est également installé.

Par exemple, lors de l'installation de `langpacks-fr` comme le montrent les exemples précédents, le paquetage `mythes-fr` ne sera installé que si le dictionnaire des synonymes `mythes` est également installé sur le système.

Si `mythes` est ensuite installé sur ce système, le paquetage `mythes-fr` sera également automatiquement installé en raison de la faible dépendance du paquetage `langpacks-fr` déjà installé.



Références

Pages de manuel `locale(7)`, `localectl(1)`, `locale.conf(5)`, `vconsole.conf(5)`, `unicode(7)` et `utf-8(7)`

Les conversions entre le nom des présentations X11 de l'environnement graphique de bureau et leur nom dans `localectl` se trouvent dans le fichier `/usr/share/X11/xkb/rules/base.lst`.

Référence des codes de langue



Note

Ce tableau peut ne pas refléter tous les langpacks disponibles sur votre système.

Utilisez `yum info langpacks-SUFFIX` pour obtenir plus d'informations sur un paquetage particulier de langpacks.

Codes de langue

Langue	Suffixe langpacks	Valeur \$LANG
Anglais (États-Unis)	en	en_US.utf8
Assamais	comme	as_IN.utf8
Bengali	bn	bn_IN.utf8
Chinois (simplifié)	zh_CN	zh_CN.UTF8
Chinois (traditionnel)	zh_TW	zh_TW.utf8
Français	FR	fr_FR.utf8
Allemand	de	de_DE.utf8
Gujarati	gu	gu_IN.utf8
Hindi	hi	hi_IN.utf8
Italien	it	it_IT.utf8
Japonais	ja	ja_JP.utf8
Kannada	kn	kn_IN.utf8
Coréen	ko	ko_KR.utf8
Malayalam	ml	ml_IN.utf8
Marathi	mr	mr_IN.utf8
Odia	ou	or_IN.utf8

Langue	Suffixe langpacks	Valeur \$LANG
Portugais (brésilien)	pt_BR	pt_BR.utf8
Pendjabi	pa	pa_IN.utf8
Russe	ru	ru_RU.utf8
Espagnol	es	es_ES.utf8
Tamoul	ta	ta_IN.utf8
Télougou	te	te_IN.utf8

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 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 enchevêtrées avec 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.

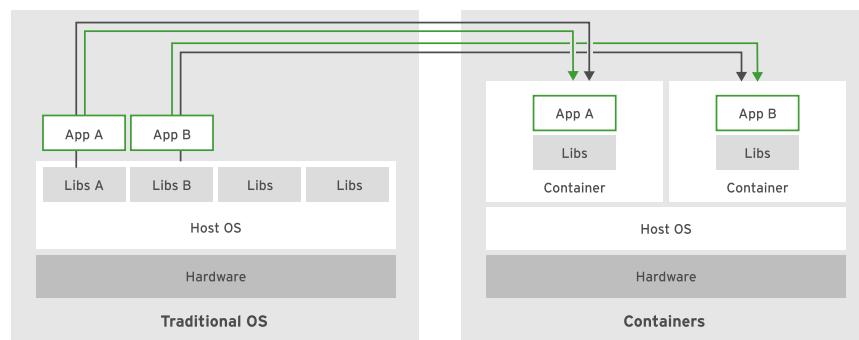


Figure 1.1: Différences entre un conteneur et un système d'exploitation

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 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-en deux.)**
- 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-en deux.)**
- 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-en deux.)**
- 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-en trois.)**
- 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-en deux.)**
- 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-en deux.)**
- 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-en deux.)**
- 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-en trois.)**
- 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, autorise 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>]
- Docker Hub [<https://hub.docker.com>]
- Red Hat Quay [<https://quay.io/>]
- 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 publics 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 :

chapitre 1 | Présentation de la technologie de conteneur

- 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-en trois.)**
 - 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. Modèle de conteneur à partir duquel un conteneur est créé.
 - 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-en trois.)**
 - 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.

► 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-en trois.)**
 - 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. Modèle de conteneur à partir duquel un conteneur est créé.
 - 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-en trois.)**
 - 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.

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 conteneurs.

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 contrôles d'intégrité définis par l'utilisateur pour surveiller les conteneurs 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

RHOCP 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 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-en trois.)**
- 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'applications à l'intérieur de ceux-ci.
 - 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-en deux.)**
- 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-en trois.)**
- 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-en deux.)**
- 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-en trois.)**

- 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'applications à l'intérieur de ceux-ci.
- 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-en deux.)**

- 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-en trois.)**

- 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-en deux.)**

- 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 pour accéder à un cluster OpenShift géré par la formation Red Hat.
- Un compte personnel et gratuit GitHub. Si vous devez vous inscrire à GitHub, reportez-vous aux instructions dans *Annexe B, 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 C, 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, sélectionnez le menu **Settings**, puis sélectionnez **Developer settings** dans le volet gauche de la page.

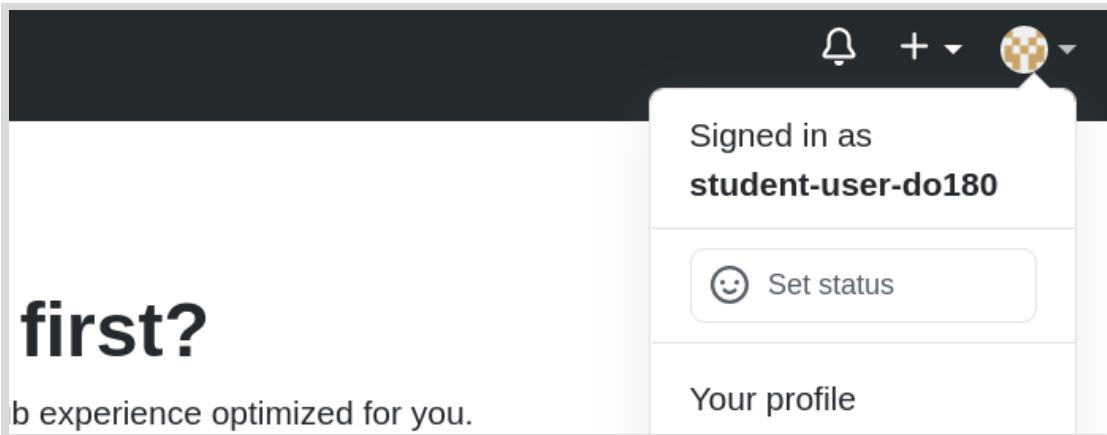


Figure 1.2: Menu User

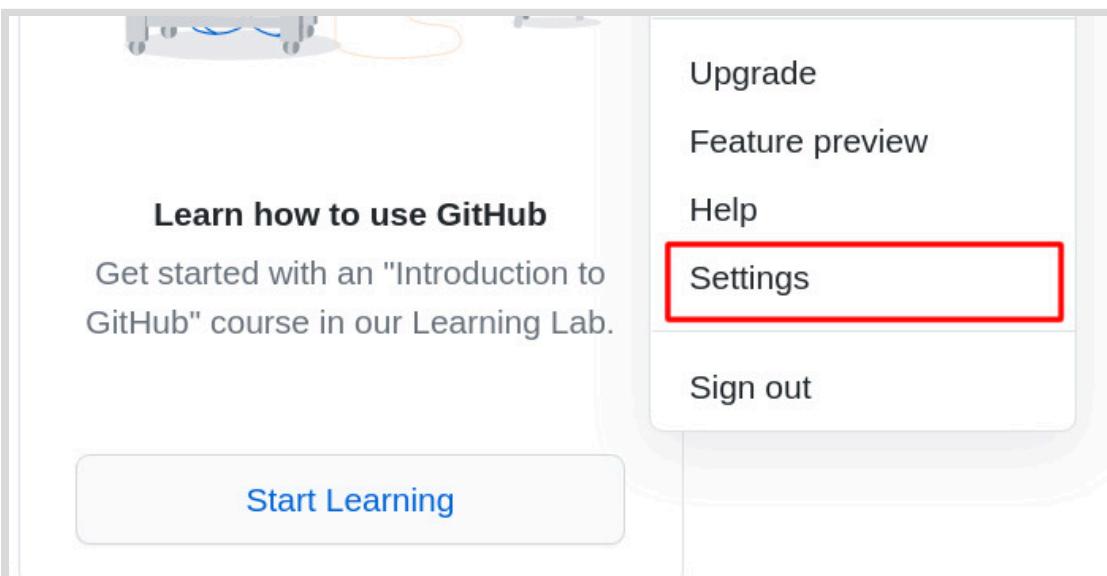


Figure 1.3: Menu Settings

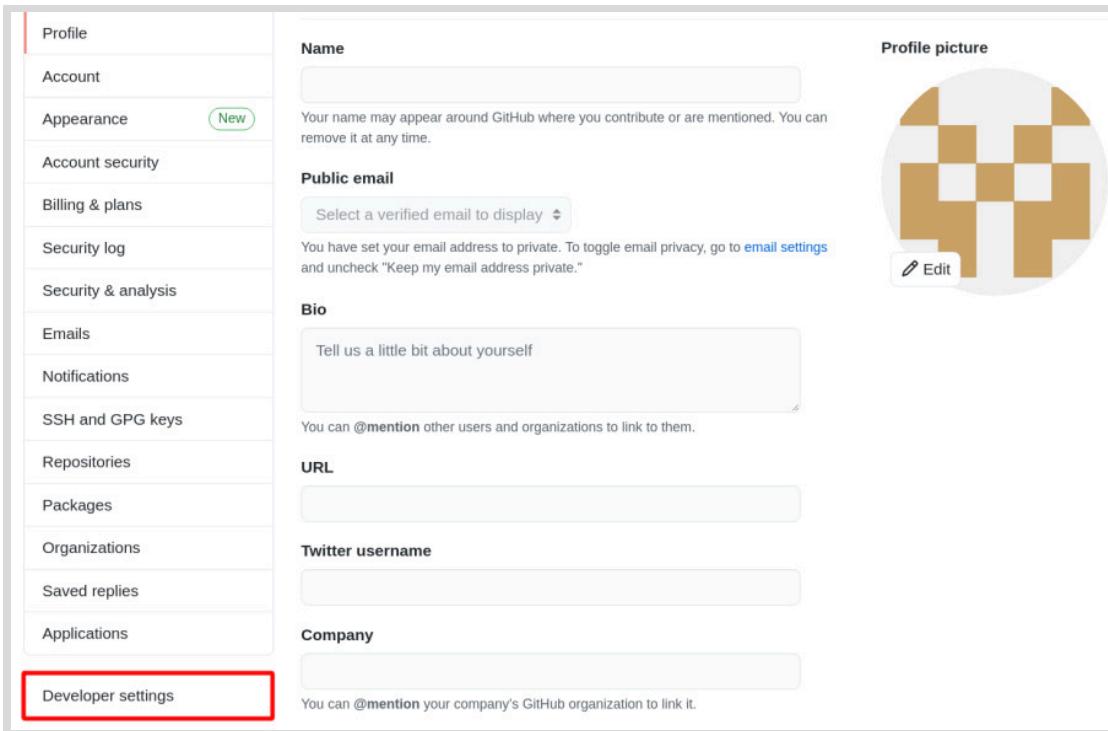


Figure 1.4: Paramètres Developer

- 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.

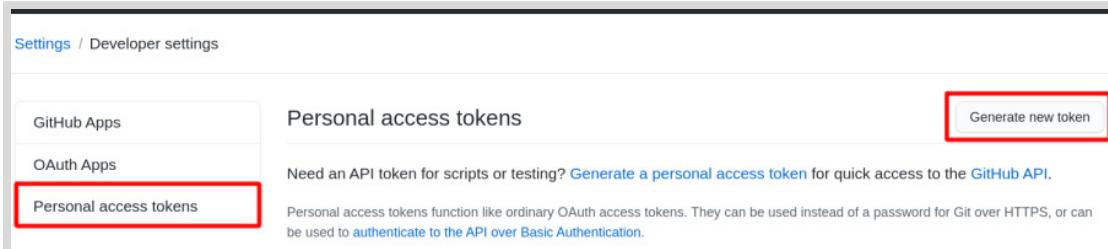


Figure 1.5: Volet Personal access token

- 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**.

New personal access token

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](#).

Note

Course DO180
What's this token for?

Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<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 1.6: 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.

Personal access tokens

Generate new token Revoke all

Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your new personal access token now. You won't be able to see it again!

✓ ghp_kgYGZwCGE1CrdovkzuzeLTWvYY6eBX2l0vCK [Copy](#) [Delete](#)

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 1.7: 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
```



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. Dans la page *Account Settings*, cliquez sur le lien *Change password*. Consultez Annexe C, *Création d'un compte Quay* pour plus de détails.

▶ 3. Configurez la machine **workstation**.

Pour les étapes suivantes, utilisez les valeurs fournies par l'environnement d'apprentissage en ligne de la formation Red Hat lorsque vous approvisionnez votre environnement d'atelier en ligne :

The screenshot shows the Red Hat OpenShift Lab Environment interface. At the top, there are tabs for 'Table of Contents', 'Course', and 'Lab Environment'. Below the tabs, there's a 'Lab Controls' section with a 'CREATE' button (highlighted in red) and a 'DELETE' button. 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 'STOP' button and an information icon. The main area is titled 'OpenShift Details' and contains a table with the following data:

	Username	Password	API Endpoint	Console Web Application	Cluster Id
Username	RHT_OCP4_DEV_USER	youruser			
Password	RHT_OCP4_DEV_PASSWORD	yourpassword			
API Endpoint	RHT_OCP4_MASTER_API		https://api.cluster.domain.example.com:6443		
Console Web Application				https://console-openshift-console.apps.cluster.domain.example.com	
Cluster Id					your-cluster-id

At the bottom, there are two rows of tables for 'workstation' and 'classroom'. Each row has an 'ACTION' dropdown and an 'OPEN CONSOLE' button. The 'workstation' row shows 'active' status, and the 'classroom' row also shows 'active' status.

Ouvrez un terminal sur la machine virtuelle **workstation** et exécutez la commande suivante. Répondez aux invites interactives avant de commencer tout autre exercice de ce cours.

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

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

- 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.

```
This script configures the connection parameters to access the OpenShift cluster  
for your lab scripts.
```

- Enter the API Endpoint: `https://api.cluster.domain.example.com:6443` ①
- Enter the Username: `youruser` ②
- Enter the Password: `yourpassword` ③
- Enter the GitHub Account Name: `yourgituser` ④
- Enter the Quay.io Account Name: `yourquayuser` ⑤

```
...output omitted...
```

① URL de l'API maîtresse de votre cluster OpenShift. Saisissez l'URL sous la forme d'une seule ligne, sans espaces ni sauts de ligne. La formation Red Hat vous procure cette information lorsque vous déployez votre atelier. Vous avez besoin de ces informations pour vous connecter au cluster et pour déployer des applications en conteneur.

② ③ Vos nom d'utilisateur et mots de passe developer OpenShift. La formation Red Hat vous procure cette information lorsque vous déployez votre atelier. Vous devez utiliser ce nom d'utilisateur et ce mot de passe pour vous connecter à OpenShift. Vous utilisez également votre nom d'utilisateur dans le cadre d'identifications telles que les noms d'hôte de route et les noms de projet, afin d'éviter toute collision avec des identifiants d'autres stagiaires partageant le même cluster OpenShift que vous.

④ ⑤ 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 B, *Création d'un compte GitHub* et Annexe C, *Création d'un compte Quay* pour obtenir des instructions sur la manière de procéder à l'inscription.

- 3.2. La commande `lab-configure` imprime toutes les informations que vous avez saisies et tente de se connecter à votre cluster OpenShift.

```
...output omitted...
```

You entered:

- API Endpoint: `https://api.cluster.domain.example.com:6443`
- Username: `youruser`
- Password: `yourpassword`
- GitHub Account Name: `yourgituser`
- Quay.io Account Name: `yourquayuser`

```
...output omitted...
```

- 3.3. 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`. La liste suivante montre un exemple d'erreur de vérification.

```
...output omitted...
```

```
Verifying your API Endpoint...
```

ERROR:

Cannot connect to an OpenShift 4 API using your URL.

Please verify your network connectivity and that the URL does not point to an OpenShift 3.x nor to a non-OpenShift Kubernetes API.

- 3.4. Si tout est OK jusqu'à présent, la commande `lab-configure` essaie d'accéder à vos comptes GitHub et Quay.IO publics.

```
...output omitted...
```

```
Verifying your GitHub account name...
```

```
Verifying your Quay.io account name...
```

```
...output omitted...
```

- 3.5. La commande `lab-configure` affiche un message d'erreur et l'application se ferme en cas de problème. Vous devez vérifier vos informations et exécuter à nouveau la commande `lab-configure`. La liste suivante montre un exemple d'erreur de vérification :

```
...output omitted...
```

```
Verifying your GitHub account name...
```

ERROR:

Cannot find a GitHub account named: invalidusername.

- 3.6. Enfin, la commande `lab-configure` vérifie que votre cluster OpenShift signale le domaine générique attendu.

```
...output omitted...
```

```
Verifying your cluster configuration...
```

```
...output omitted...
```

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

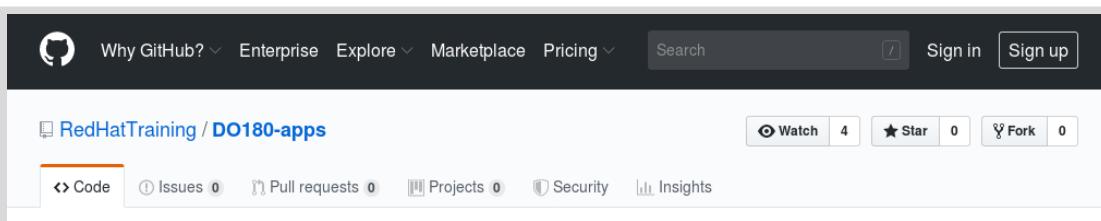
```
...output omitted...

Saving your lab configuration file...

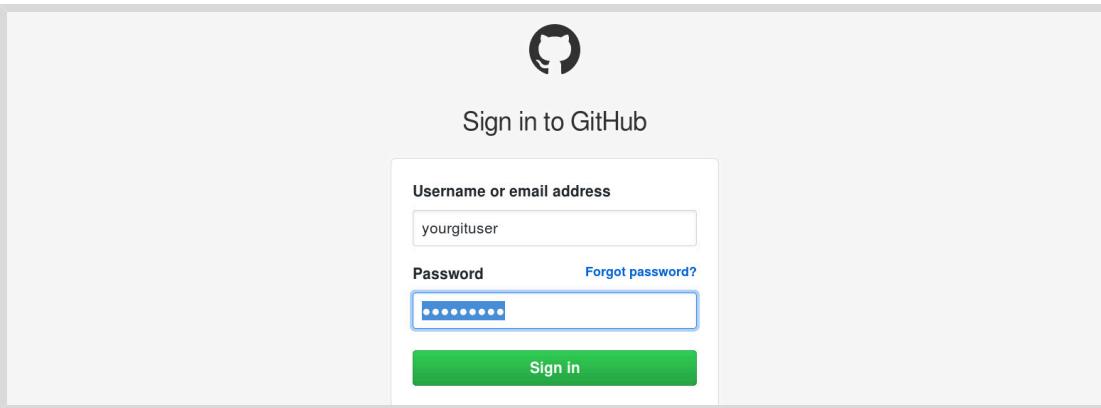
All fine, lab config saved. You can now proceed with your exercises.

If you need to modify the configuration, rerun this or directly modify the values
in /usr/local/etc/ocp4.config.
```

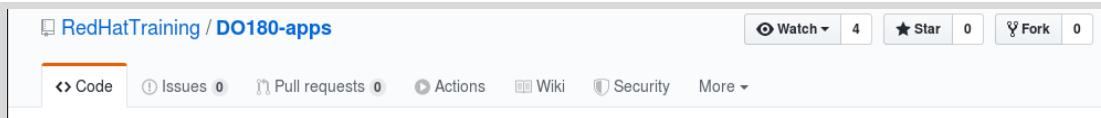
- 3.8. Si aucune erreur n'est survenue lors de l'enregistrement de votre configuration, vous êtes presque prêt à lancer l'un des exercices de ce cours. En cas d'erreur, n'essayez pas de démarrer un exercice tant que vous ne pouvez pas exécuter la commande `lab-configure` avec succès.
- 4. Scinez les exemples d'applications de ce cours dans votre compte GitHub personnel. Effectuez les étapes suivantes :
- 4.1. Ouvrez un navigateur Web et accédez à <https://github.com/RedHatTraining/DO180-apps>. Si vous n'êtes pas connecté à GitHub, cliquez sur **Sign in** dans le coin supérieur droit.



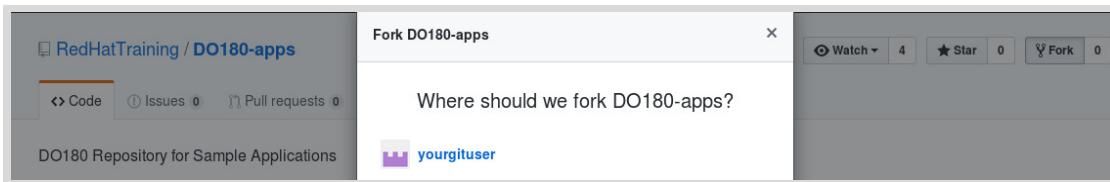
- 4.2. Connectez-vous à GitHub à l'aide de votre nom d'utilisateur personnel et de votre mot de passe.



- 4.3. Accédez au référentiel `RedHatTraining/DO180-apps`, puis cliquez sur **Fork** dans le coin supérieur droit.



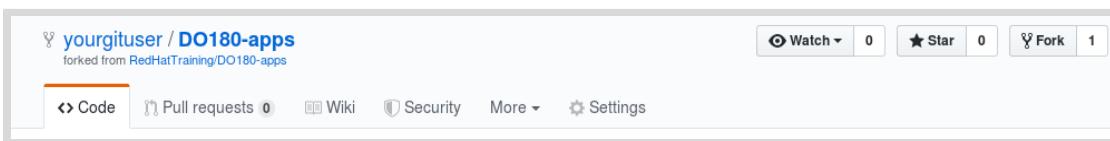
- 4.4. Dans la fenêtre Fork `DO180-apps`, cliquez sur `yourgituser` pour sélectionner votre projet GitHub personnel.



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 **DO180-apps** lorsque vous scindez le référentiel.

- 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 :

- 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...
```

- Vérifiez que `/home/user/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
```

- 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
```

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

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

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

```
[student@workstation D0180-apps]$ git push --set-upstream origin testbranch
Username for https://github.com: ①
Password for https://yourgituser@github.com: ②
...output omitted...
```

- ① Saisissez votre nom d'utilisateur GitHub
- ② 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/user/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 ~]$
```

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 créez 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 E, 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

- Déploiement d'un serveur de base de données (avec exercice guidé)
- Utilisation des conteneurs sans racine (avec exercice guidé)

Atelier

- Crédit de services en conteneur

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...
```

chapitre 2 | Création de services en conteneur

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 :

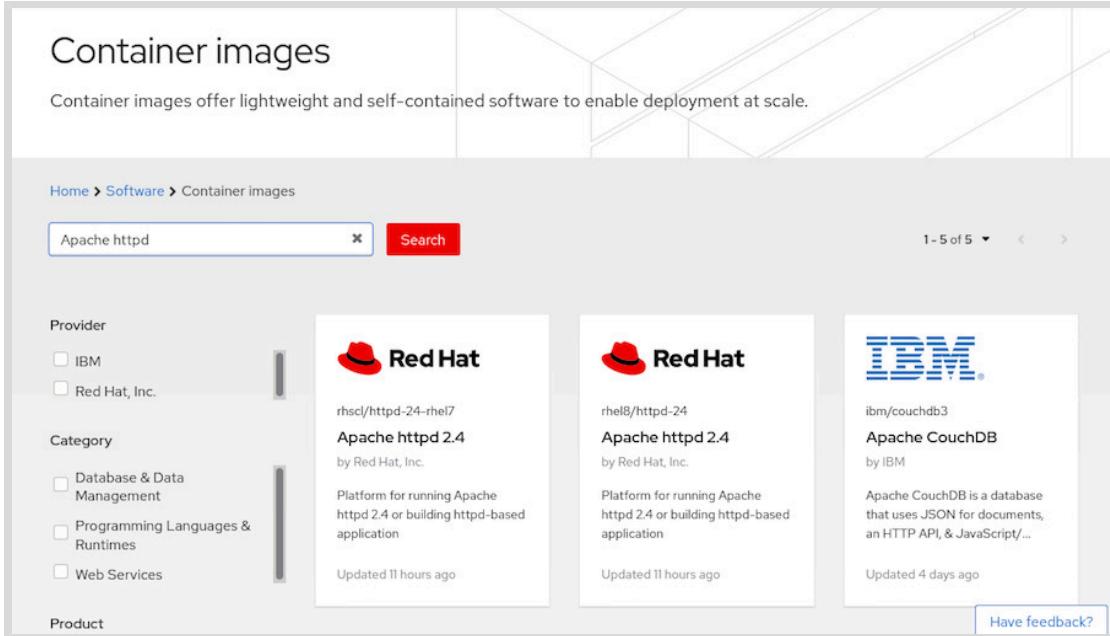


Figure 2.1: Page d'accueil de Red Hat Container Catalog

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 :

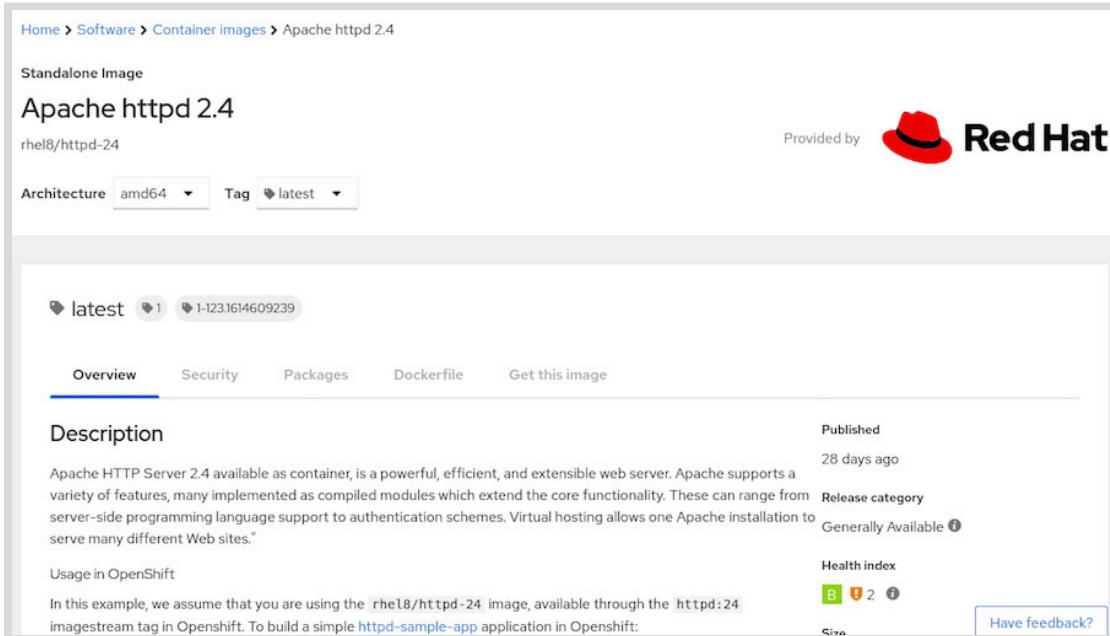


Figure 2.2: Page d'image d'aperçu Apache httpd 2.4 (rhel8/httpd-24)

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.

chapitre 2 | Création de services en conteneur

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.

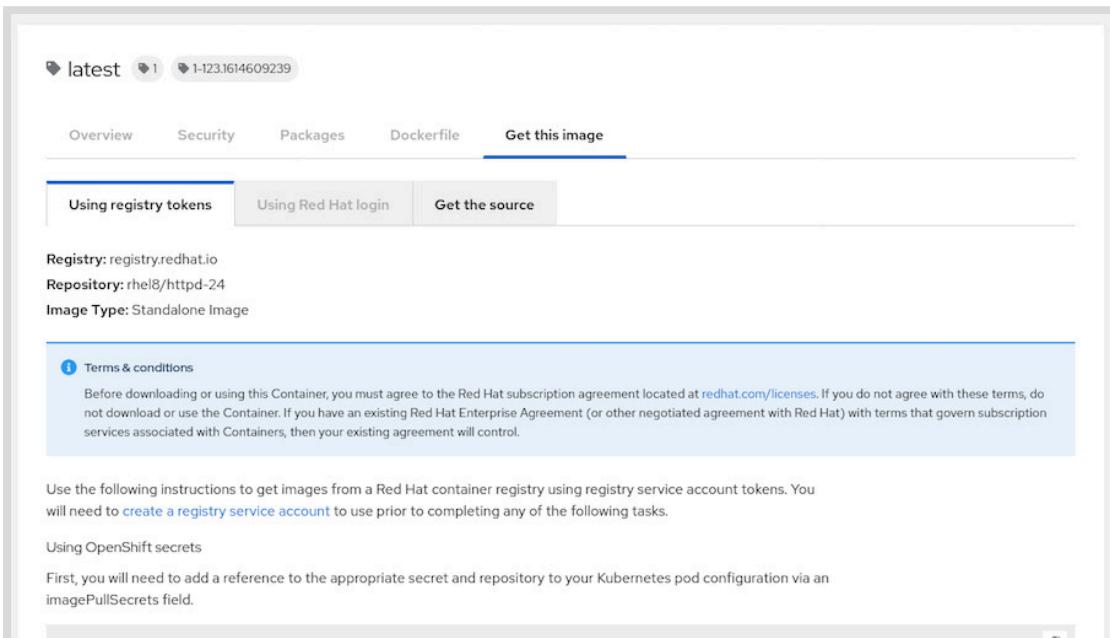


Figure 2.3: Dernière page d'image Apache httpd 2.4 (rhel8/httpd-24)

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 D, 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
`2d37682eb33a`70330259d6798bdfdc37921367f56b9c2a97339d84faa3446a03
```

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.

Utilisation de conteneurs sans racine

Résultats

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

- Expliquer les différences entre l'exécution de conteneurs avec et sans racine.
- Décrire les avantages et les inconvénients de chaque cas.
- Exécuter en tant que conteneurs avec et sans racine avec Podman.

Évolution de l'utilisation des conteneurs

Si vous exécutez des conteneurs depuis un certain temps, il y a de fortes chances que vous les exécutez en tant qu'utilisateur privilégié. Par le passé, les outils de création de conteneur exigeaient que les moteurs d'exécution s'exécutent en tant que root, et un accès privilégié était nécessaire pour créer des ressources, telles que des interfaces réseau.

Du point de vue de la sécurité, fournir ce niveau d'accès est une mauvaise pratique. Vous devez toujours exécuter des logiciels avec des priviléges aussi limités que possible. Lorsqu'un bogue de sécurité est exploité, que ce soit sur le moteur d'exécution ou sur l'application proprement dite, l'impact est minimisé.

Une meilleure pratique consiste à mettre les applications en conteneur de telle sorte qu'elles n'ont pas besoin d'un utilisateur privilégié pour s'exécuter. À la place, ces applications doivent utiliser un utilisateur bien connu.

De nombreuses images de conteneurs communautaires, telles que celles disponibles dans docker.io, ont toujours besoin de root pour s'exécuter.

Certains outils, tels que Podman et Red Hat OpenShift, exécutent par défaut des conteneurs sans racine. Docker annonce que le mode sans racine est généralement disponible dans la version 20.10.

Avantages des conteneurs sans racine

Les conteneurs sans racine sont un nouveau concept de conteneurs qui n'ont pas besoin de priviléges root pour s'exécuter. Les conteneurs sans racine sont avantageux du point de vue de la sécurité pour plusieurs raisons, notamment :

- Permet que le code s'exécute dans un conteneur sans racine doté de priviléges root, sans avoir à s'exécuter en tant qu'utilisateur root de l'hôte.
- Ajoute une nouvelle couche de sécurité ; si le moteur du conteneur est compromis, l'attaquant n'aura pas de priviléges root sur l'hôte.
- Permet à plusieurs utilisateurs sans privilège d'exécuter des conteneurs sur la même machine.
- Permet l'isolement dans des conteneurs imbriqués.

Malgré tous ces avantages, l'exécution de conteneurs sans racine présente plusieurs limitations, notamment :

chapitre 2 | Création de services en conteneur

- Fonctionnalités supprimées
- Liaison aux ports inférieurs à 1024
- Montage de volumes d'autres contenus

Il existe une liste détaillée des inconvénients liés aux conteneurs sans racine dans <https://github.com/containers/podman/blob/master/rootless.md>.

Comprendre les conteneurs sans racine

Pour comprendre le fonctionnement des conteneurs sans racine, prenez en compte les concepts suivants.

Espaces de noms User

Les conteneurs utilisent des espaces de noms Linux pour s'isoler de l'hôte sur lequel ils s'exécutent. L'espace de noms User est notamment utilisé pour créer des conteneurs sans racine. Cet espace de noms met en correspondance les ID d'utilisateur et de groupe de sorte qu'un processus à l'intérieur de l'espace de noms puisse apparaître sous un ID différent.

Les conteneurs sans racine utilisent l'espace de noms d'utilisateurs User pour que le code de l'application semble s'exécuter en tant que root. Cependant, du point de vue de l'hôte, les autorisations sont limitées à celles d'un utilisateur standard. Si un attaquant parvient à échapper l'espace de noms d'utilisateur sur l'hôte, il ne disposera que des fonctionnalités d'un utilisateur normal sans autorisation.

Mise en réseau

Pour permettre une mise en réseau adéquate à l'intérieur d'un conteneur, un périphérique Ethernet virtuel est créé. Cela pose un problème pour les conteneurs sans racine, car seul un véritable utilisateur root dispose des priviléges nécessaires pour créer ce périphérique et d'autres périphériques similaires.

Sur un conteneur sans racine, la mise en réseau est généralement gérée par Slirp. Elle fonctionne en créant une fourche dans les espaces de noms d'utilisateur et de réseau du conteneur et en créant un périphérique TAP qui devient la route par défaut. Ensuite, le descripteur de fichier du périphérique est transmis au parent, qui s'exécute dans l'espace de noms de réseau par défaut et peut désormais communiquer avec le conteneur et Internet.

Storage (stockage)

Par défaut, les moteurs de conteneur utilisent un pilote spécial appelé Overlay2 (ou Overlay) pour créer un système de fichiers en couches qui est efficace à la fois en termes de capacité et de performances. Cela ne peut pas être fait avec des conteneurs sans racine, car la plupart des distributions Linux n'autorisent pas le montage des systèmes de fichiers superposés dans les espaces de noms d'utilisateur.

Pour les conteneurs sans racine, la solution consiste à créer un pilote de stockage. FUSE-OverlayFS est une implémentation d'espace utilisateur d'Overlay, qui est plus efficace que le pilote de stockage VFS utilisé avant et peut s'exécuter au sein d'espaces de noms d'utilisateurs.



Références

Page de manuel user_namespaces(7)

https://man7.org/linux/man-pages/man7/user_namespaces.7.html

Utilisation de Podman dans un environnement sans racine

https://github.com/containers/podman/blob/master/docs/tutorials/rootless_tutorial.md

► Exercice guidé

Découverte des conteneurs avec et sans racine

Dans cet exercice, vous allez comprendre les différences entre l'exécution de conteneurs en tant que racine et leur exécution sans racine.

Résultats

Vous devez pouvoir voir les UUID des processus s'exécutant à l'intérieur des conteneurs.

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 permet de s'assurer que vous disposez des outils nécessaires pour effectuer cet exercice.

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

Instructions

- ▶ 1. Exécutez un conteneur en tant qu'utilisateur root et vérifiez l'UID d'un processus en cours d'exécution à l'intérieur de celui-ci.
 - 1.1. Démarrez un conteneur avec sudo à partir de l'image UBI 8 Red Hat.

```
[student@workstation ~]$ sudo podman run --rm --name asroot -ti \
> registry.access.redhat.com/ubi8:latest /bin/bash
Trying to pull registry.access.redhat.com/ubi8:latest...
Getting image source signatures
...output omitted...
[root@f95d16108991 /]# whoami
root
[root@f95d16108991 /]# id
uid=0(root) gid=0(root) groups=0(root)
```

- 1.2. Démarrez un processus `sleep` à l'intérieur du conteneur.

```
[root@f95d16108991 /]# sleep 1000
```

- 1.3. Sur un nouveau terminal, exécutez `ps` pour rechercher le processus.

```
[student@workstation ~]$ sudo ps -ef | grep "sleep 1000"
root      3137      3117  0 10:18 pts/0    00:00:00 /usr/bin/coreutils --
coreutils-prog-shebang=sleep /usr/bin/sleep 1000
```

1.4. Quittez le conteneur.

```
[root@f95d16108991 /]# sleep 1000
^C
[root@f95d16108991 /]# exit
exit
[student@workstation ~]$
```

► 2. Exécutez un conteneur en tant qu'utilisateur standard et vérifiez l'UID d'un processus en cours d'exécution à l'intérieur de celui-ci.

2.1. Démarrez un autre conteneur à partir de l'UBI 8 Red Hat en tant qu'utilisateur standard.

```
[student@workstation ~]$ podman run --rm --name asuser -ti \
> registry.access.redhat.com/ubi8:latest /bin/bash
Trying to pull registry.access.redhat.com/ubi8:latest...
Getting image source signatures
...output omitted...
[root@d289dcccd5285 /]# whoami
root
[root@d289dcccd5285 /]# id
uid=0(root) gid=0(root) groups=0(root)
```

2.2. Démarrez un processus `sleep` à l'intérieur du conteneur.

```
[root@d289dcccd5285 /]# sleep 2000
```

2.3. Sur un nouveau terminal, exécutez `ps` pour rechercher le processus.

```
[student@workstation ~]$ sudo ps -ef | grep "sleep 2000" | grep -v grep
student      3345      3325  0 10:24 pts/0    00:00:00 /usr/bin/coreutils --
coreutils-prog-shebang=sleep /usr/bin/sleep 2000
```

2.4. Quittez le conteneur.

```
[root@d289dcccd5285 /]# sleep 2000
^C
[root@d289dcccd5285 /]# exit
exit
[student@workstation ~]$
```

Fin

Sur la machine `workstation`, utilisez la commande `lab` pour mettre fin à l'exercice. Il s'agit d'une étape importante pour vous assurer que les ressources des exercices précédents n'ont pas d'incidence sur les exercices à venir.

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

La section est maintenant terminée.

► Open Lab

Création de services en conteneur

Résultats

Vous serez en mesure de démarrer et de personnaliser un conteneur en utilisant une image conteneur.

Avant De Commencer

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

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

Instructions

1. Utilisez l'image `quay.io/redhattraining/httpd-parent` avec la balise `2.4` pour démarrer un nouveau conteneur nommé `httpd-basic` en arrière-plan. Acheminez le port `8080` sur l'hôte vers le port `80` dans le conteneur. Utilisez l'option `-p` de la commande `podman` et définissez la valeur de l'option sur `8080:80`.
2. Testez que le serveur HTTP Apache est en cours d'exécution à l'intérieur du conteneur `httpd-basic`.
3. Personnalisez le conteneur `httpd-basic` pour afficher `Hello World` comme message. Le message du conteneur est stocké dans le fichier `/var/www/html/index.html`.

Évaluation

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

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

Fin

Sur la machine `workstation`, exécutez le script `lab container-review finish` pour mettre fin à l'atelier.

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

L'atelier est maintenant terminé.

► Solution

Création de services en conteneur

Résultats

Vous serez en mesure de démarrer et de personnaliser un conteneur en utilisant une image conteneur.

Avant De Commencer

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

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

Instructions

1. Utilisez l'image `quay.io/redhattraining/httpd-parent` avec la balise `2.4` pour démarrer un nouveau conteneur nommé `httpd-basic` en arrière-plan. Acheminez le port `8080` sur l'hôte vers le port `80` dans le conteneur. Utilisez l'option `-p` de la commande `podman` et définissez la valeur de l'option sur `8080:80`.
 - 1.1. Utilisez la commande `podman run` pour démarrer un conteneur. Ajoutez l'option `-d` pour le démarrer en arrière-plan et ajoutez l'option `-p` pour mapper le port `8080` sur l'hôte sur le port `80` dans le conteneur.

```
[student@workstation ~]$ podman run -d -p 8080:80 --name httpd-basic \
> quay.io/redhattraining/httpd-parent:2.4
...output omitted...
Copying blob 743f2d6...output omitted...
Copying blob c92eb69...output omitted...
Copying blob 2211b05...output omitted...
...output omitted...
Copying blob aed1801...output omitted...
Writing manifest to image destination
Storing signatures
`b51444e3b1d7`aaaf94b3a4a54485d76a0a094cbfac89c287d360890a3d2779a5a
```

Cette commande démarre le serveur HTTP Apache en arrière-plan et vous renvoie l'invite Bash.

2. Testez que le serveur HTTP Apache est en cours d'exécution à l'intérieur du conteneur `httpd-basic`.
 - 2.1. À partir de la machine `workstation`, essayez d'accéder à `http://localhost:8080` au moyen de n'importe quel navigateur Web.

Le message `Hello from the httpd-parent container!` s'affiche. Ce message se trouve sur la page `index.html` du conteneur de serveur HTTP Apache exécuté sur la machine `workstation`.

chapitre 2 | Création de services en conteneur

```
[student@workstation ~]$ curl http://localhost:8080
Hello from the httpd-parent container!
```

3. Personnalisez le conteneur `httpd-basic` pour afficher Hello World comme message. Le message du conteneur est stocké dans le fichier `/var/www/html/index.html`.

- 3.1. Démarrez une session Bash à l'intérieur du conteneur.

Exécutez la commande suivante :

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

- 3.2. À partir de la session Bash, vérifiez le fichier `index.html` sous le répertoire `/var/www/html` au moyen de la commande `ls -la`.

```
bash-4.4# ls -la /var/www/html
total 4
drwxr-xr-x. 2 root root 24 Jun 12 11:58 .
drwxr-xr-x. 4 root root 33 Jun 12 11:58 ..
-rw-r--r--. 1 root root 39 Jun 12 11:58 index.html
```

- 3.3. Modifiez le fichier `index.html` pour qu'il contienne le texte Hello World, en remplaçant l'ensemble du contenu existant.

À partir de la session Bash dans le conteneur, exécutez la commande suivante :

```
bash-4.4# echo "Hello World" > /var/www/html/index.html
```

- 3.4. Essayez d'accéder à `http://localhost:8080` à nouveau et vérifiez que la page Web a été mise à jour.

```
bash-4.4# exit
exit
[student@workstation ~]$ curl http://localhost:8080
Hello World
```

Évaluation

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

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

Fin

Sur la machine `workstation`, exécutez le script `lab container-review finish` pour mettre fin à l'atelier.

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

L'atelier est maintenant terminé.

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 conteneurs. La figure suivante affiche un résumé des sous-commandes les plus couramment utilisées qui modifient l'état du conteneur et de l'image :

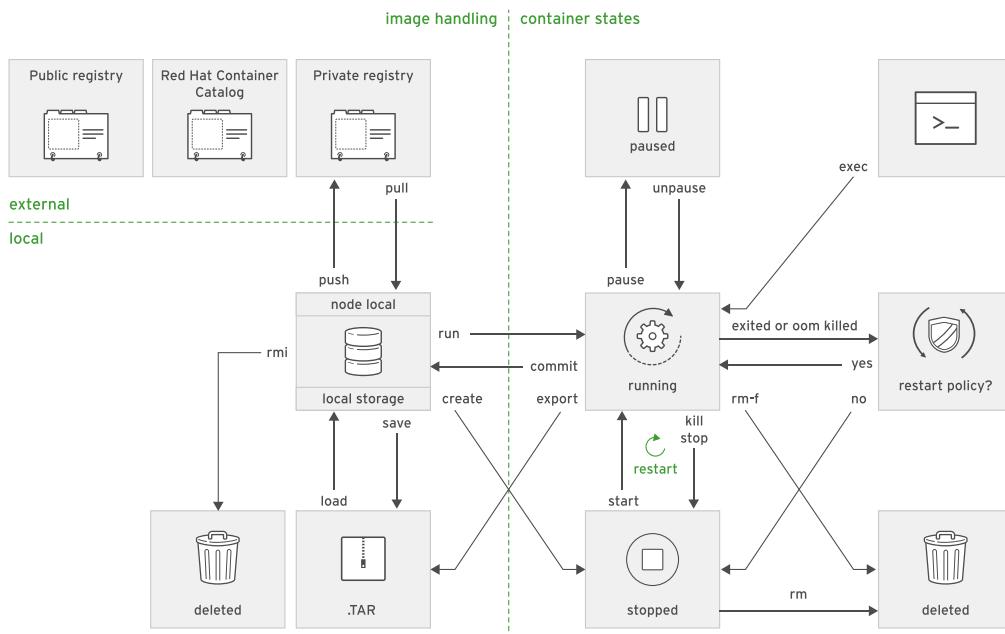


Figure 3.1: Gestion des sous-commandes par Podman

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

chapitre 3 | Gestion des conteneurs

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

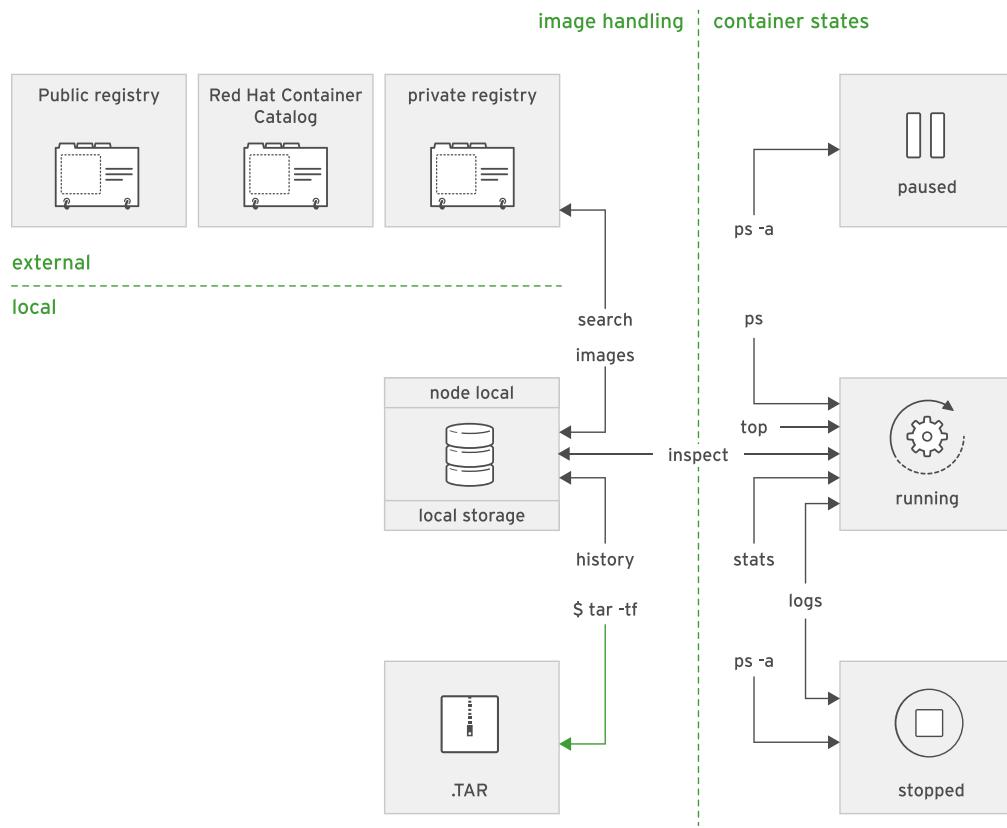


Figure 3.2: Sous-commandes de requête Podman

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.

Podman fournit les commandes suivantes pour gérer les conteneurs :

- `podman ps` : cette commande liste les conteneurs en cours d'exécution :

chapitre 3 | Gestion des conteneurs

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

- ❶ Chaque ligne décrit les informations relatives au conteneur.

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.

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

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

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

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.

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

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
4829d82fbfff registry.redhat.io/rhel8/httpd-24 "/usr/bin/run-http..." ...ago
Exited (0)... my-htt...
```

**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-htt...-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	Abandon du signal d'abandon (3)
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 l'alarme (2)
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
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
SIGTTOU	22,22,27	Stop	Sortie tty pour le processus d'arrière-plan

chapitre 3 | Gestion des conteneurs

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


Note
Terme

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-httplibd-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.

- La commande **podman rm** supprime un conteneur et ignore son état et son système de fichiers.

```
[user@host ~]$ podman rm my-httplibd-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
86162c906b44f4cb63ba2e3386554030dcba6abedbce9e9fcad60aa9f8b2d5d4
```

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 D, *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: '^$')
  MYSQL_PASSWORD (regex: '^@[a-zA-Z0-9_~!@#$%^&*()-=<>, .?;:|]$')
  MYSQL_DATABASE (regex: '^$')
`Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^@[a-zA-Z0-9_~!@#$%^&*()-=<>, .?;:|]$')
Or both.
Optional Settings:
```

```
...output omitted...
```

For more information, see <https://github.com/sclorg/mysql-container>



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.

```
a49dba9ff17f2b5876001725b581fdd331c9ab8b9eda21cc2a2899c23f078509
```

- ▶ 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 \
> /home/student/D0180/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
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, telles que les bases de données. Pour prendre en charge les applications de ce type, l'administrateur doit fournir un conteneur avec du stockage persistant.

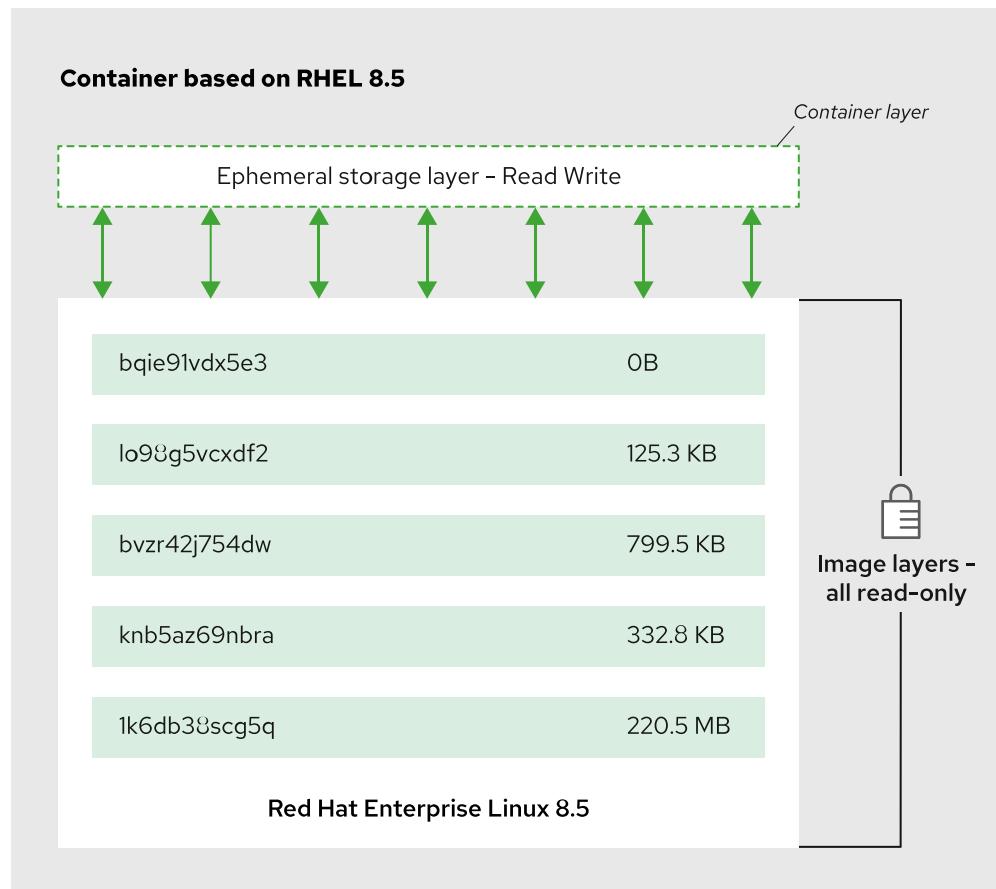


Figure 3.3: Couches du conteneur

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 un 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.

► 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 -pv /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
```

- 1.5. 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 D, 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 ...output omitted...registry.redhat.io/rhel8/mysql-80:1...output
omitted...
...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 de données MySQL stocke les données consignées dans le répertoire /var/lib/mysql/data.

- 2.4. Vérifiez que le conteneur a été correctement démarré.

chapitre 3 | Gestion des conteneurs

```
[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 127.0.0.1:35134
```

```
...output omitted...
<h1>Red Hat Enterprise Linux <strong>Test Page</strong></h1>
<div class="content">
<div class="content-middle">
<p>This page is used to test the proper operation of the HTTP server after
it has been installed. If you can read this page, it means that the HTTP server
installed at this site is working properly.</p>
...output omitted...
```

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 D, 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 mysqlDb-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 ...output omitted...
Copying blob sha256:1c9f515...output omitted...
72.70 MB / ? [=====] 5s
Copying blob sha256:1d2c4ce...output omitted...
1.54 KB / ? [=====] 0s
Copying blob sha256:f1e961f...output omitted...
6.85 MB / ? [=====] 0s
```

chapitre 3 | Gestion des conteneurs

```
Copying blob sha256:9f1840c...output omitted...
62.31 MB / ? [-----] 7s
Copying config sha256:60726...output omitted...
6.85 KB / 6.85 KB [=====] 0s
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  mysqlldb-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 mysqlldb-port \
> mysql -uroot items -e "SELECT * FROM Item"
-----
| id | description      | done |
-----
| 1  | Pick up newspaper |  0  |
| 2  | Buy groceries     |  1  |
-----
```

chapitre 3 | Gestion des conteneurs

- 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 mysqlbdb-port /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
```

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

1. Créez le répertoire `/home/student/local/mysql` avec les permissions et le contexte SELinux corrects.
2. Déployez une instance de conteneur MySQL à l'aide des caractéristiques suivantes :
 - *Name* : `mysql-1`
 - *Run as daemon* : oui
 - *Volume* : depuis le dossier hôte `/home/student/local/mysql` vers le dossier de conteneur `/var/lib/mysql/data`
 - *Container image* : `registry.redhat.io/rhel8/mysql-80:1`
 - *Port forward* : oui, du port hôte 13306 au port de conteneur 3306
 - *Environment variables* :
 - `MYSQL_USER`: `user1`.
 - `MYSQL_PASSWORD`: `mypa55`.
 - `MYSQL_DATABASE`: `items`.
 - `MYSQL_ROOT_PASSWORD`: `r00tpa55`.
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 les caractéristiques suivantes :
 - *Name* : mysql-2
 - *Run as a daemon* : oui
 - *Volume* : depuis le dossier hôte /home/student/local/mysql vers le dossier de conteneur /var/lib/mysql/data
 - *Container image* : registry.redhat.io/rhel8/mysql-80:1
 - *Port forward* : oui, du port hôte 13306 au port de conteneur 3306
 - *Environment variables* :
 - MYSQL_USER: user1.
 - MYSQL_PASSWORD: mypa55.
 - MYSQL_DATABASE: items.
 - MYSQL_ROOT_PASSWORD: r00tpa55.
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

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 -pv /home/student/local/mysql  
mkdir: created directory '/home/student/local/mysql'
```

- 1.2. 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(/.*)?'
```

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

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

- 1.4. 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
```

2. Déployez une instance de conteneur MySQL à l'aide des caractéristiques suivantes :

chapitre 3 | Gestion des conteneurs

- *Name* : mysql-1
- *Run as daemon* : oui
- *Volume* : depuis le dossier hôte /home/student/local/mysql vers le dossier de conteneur /var/lib/mysql/data
- *Container image* : registry.redhat.io/rhel8/mysql-80:1
- *Port forward* : oui, du port hôte 13306 au port de conteneur 3306
- *Environment variables* :
 - 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 ...output omitted...
...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
```

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 instruction 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
616azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cdd
```

5. Créez un conteneur avec les caractéristiques suivantes :

- *Name* : mysql-2
- *Run as a daemon* : oui
- *Volume* : depuis le dossier hôte /home/student/local/mysql vers le dossier de conteneur /var/lib/mysql/data
- *Container image* : registry.redhat.io/rhel8/mysql-80:1
- *Port forward* : oui, du port hôte 13306 au port de conteneur 3306
- *Environment variables* :
 - MYSQL_USER: user1.

chapitre 3 | Gestion des conteneurs

- MYSQL_PASSWORD: mypa55.
- MYSQL_DATABASE: items.
- MYSQL_ROOT_PASSWORD: r00tpa55.

5.1. Créez et démarrez le conteneur.

```
[student@workstation ~]$ podman run --name mysql-2 \
> -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
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
```

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 |
```

chapitre 3 | Gestion des conteneurs

```
| 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.

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

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
```

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

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 <number></code>	Limite le nombre d'images listées par registre.
<code>--filter <filter=value></code>	Filtrer la sortie en fonction des conditions fournies. Les filtres pris en charge sont les suivants : <code>stars=<number></code> : afficher uniquement les images ayant au moins ce nombre d'étoiles. <code>is-automated=<true false></code> : afficher uniquement les images générées automatiquement. <code>is-official=<true false></code> : afficher uniquement les images marquées comme officielles.
<code>--tls-verify <true false></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 où elles apparaissent dans le fichier de configuration.

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

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
registry.redhat.io	nginx/1.16
	mysql/8.0
	httpd/2.4
quay.io	mysql/5.7
	httpd/2.4

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-en deux.)
 - a. podman search registry.redhat.io/mysql
 - b. podman images
 - c. podman pull mysql
 - d. 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. There is no podman command available to search for tags.

► 3. Quelles sont les deux commandes permettant d'extraire l'image httpd avec la balise

2.4 ? (Choisissez-en deux.)

- 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 ?

```
podman pull registry.redhat.io/httpd:2.4
```

```
podman pull quay.io/mysql:8.0
```

- a. quay.io/httpd:2.4 registry.redhat.io/mysql:8.0
- b. registry.redhat.io/httpd:2.4 registry.redhat.io/mysql:8.0
- c. registry.redhat.io/httpd:2.4 No image will be downloaded for mysql.
- d. quay.io/httpd:2.4 No image will be downloaded for mysql.

► Solution

Utilisation des registres

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
registry.redhat.io	nginx/1.16
	mysql/8.0
	httpd/2.4
quay.io	mysql/5.7
	httpd/2.4

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-en deux.)
 - a. `podman search registry.redhat.io/mysql`
 - b. `podman images`
 - c. `podman pull mysql`
 - d. `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. There is no podman command available to search for tags.

► 3. Quelles sont les deux commandes permettant d'extraire l'image httpd avec la balise 2.4 ? (Choisissez-en deux.)

- 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 ?

```
podman pull registry.redhat.io/httpd:2.4
```

```
podman pull quay.io/mysql:8.0
```

- a. quay.io/httpd:2.4 registry.redhat.io/mysql:8.0
- b. registry.redhat.io/httpd:2.4 registry.redhat.io/mysql:8.0
- c. registry.redhat.io/httpd:2.4 No image will be downloaded for mysql.
- d. quay.io/httpd:2.4 No image will be downloaded for 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. Publier (*pousser*) l'image de conteneur vers un registre d'images.



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` :

chapitre 4 | Gestion des images de conteneur

```
[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
...output omitted...
Writing manifest to image destination
Storing signatures
`3a6baecaff2b`4e8c53b026e04847dda5976b773ade1a3a712b1431d60ac5915d
```

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`.
- 2.1. Accédez au shell du conteneur en utilisant la commande `podman exec`, puis créez une page HTML.

chapitre 4 | Gestion des images de conteneur

```
[student@workstation ~]$ podman exec -it official-httdp /bin/bash  
bash-4.4# echo "DO180 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
```

Le résultat devrait être semblable à ce qui suit :

```
DO180 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.

- 4. Créez une nouvelle image avec les modifications créées par le conteneur en cours d'exécution.

4.1. Arrêtez le conteneur official-httdp.

chapitre 4 | Gestion des images de conteneur

```
[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  
Skipping fetch of repeat blob sha256:071d8bd765171080d01682844524be57ac9883e...  
...output omitted...  
Copying blob sha256:1e19be875ce6f5b9dece378755eb9df96ee205abfb4f165c797f59a9...  
15.00 KB / 15.00 KB [=====] 0s  
Copying config sha256:8049dc2e7d0a0b1a70fc0268ad236399d9f5fb686ad4e31c7482cc...  
2.99 KB / 2.99 KB [=====] 0s  
Writing manifest to image destination  
Storing signatures  
'31c3ac78e9d4`137c928da23762e7d32b00c428eb1036cab1caeeb399befef2a23
```

- 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	31c3ac78e9d4
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
Copying blob sha256:071d8bd765171080d01682844524be57ac9883e53079b6ac66707e19...
200.44 MB / 200.44 MB [=====] 1m38s
...output omitted...
Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb3...
2.99 KB / 2.99 KB [=====] 0s
Writing manifest to image destination
Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb3...
0 B / 2.99 KB [-----] 0s
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
31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb3
```

► 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.
 - *Name* : `official-nginx`
 - *Run as daemon* : oui
 - *Container image* : `nginx`
 - *Port forward* : du port hôte 8080 au port 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 :
 - Nom de l'image : `do180/mynginx`
 - Balise de l'image : `v1.0-SNAPSHOT`
 - Nom de l'auteur : *otre nom*
5. Démarrez un nouveau conteneur au moyen de l'image Nginx mise à jour, conformément aux spécifications figurant dans la liste suivante.

chapitre 4 | Gestion des images de conteneur

- *Name* : official-nginx-dev
 - *Run as daemon* : oui
 - *Container image* : do180/mynginx:v1.0-SNAPSHOT
 - *Port forward* : du port hôte 8080 au port 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 :
- Nom de l'image : `do180/mynginx`
 - Balise de l'image : `v1.0`
 - Nom de l'auteur : *votre nom*
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 :
- Container name : `my-nginx`
 - Run as daemon : oui
 - Container image : `do180/mynginx:v1.0`
 - Port forward : depuis le port d'hôte 8280 vers le 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

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.

- 1.1. 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
```

- 1.2. 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

2. Démarrez un nouveau conteneur au moyen de l'image Nginx, conformément aux spécifications figurant dans la liste suivante.

- *Name* : `official-nginx`
- *Run as daemon* : oui
- *Container image* : `nginx`

chapitre 4 | Gestion des images de conteneur

- *Port forward* : du port hôte 8080 au port 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/redhattraining/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 `d0180/mynginx` et la balise `v1.0-SNAPSHOT`. Utilisez les spécifications suivantes :

- Nom de l'image : `d0180/mynginx`
- Balise de l'image : `v1.0-SNAPSHOT`
- Nom de l'auteur : *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.

chapitre 4 | Gestion des images de conteneur

```
[student@workstation ~]$ podman commit -a 'Your Name' \
> official-nginx do180/mynginx:v1.0-SNAPSHOT
Getting image source signatures
...output omitted...
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.

- *Name* : **official-nginx-dev**
- *Run as daemon* : oui
- *Container image* : **do180/mynginx:v1.0-SNAPSHOT**
- *Port forward* : du port hôte 8080 au port conteneur 80.

- 5.1. Sur **workstation**, utilisez la commande `podman run` pour créer un conteneur nommé **official-nginx-dev**.

```
[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.

chapitre 4 | Gestion des images de 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  
DO180 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 :

- Nom de l'image : `do180/mynginx`
- Balise de l'image : `v1.0`
- Nom de l'auteur : *votre nom*

7.1. Utilisez la commande ` `podman sto` p 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.

```
[student@workstation ~]$ podman commit -a 'Your Name' \  
> official-nginx-dev do180/mynginx:v1.0  
Getting image source signatures  
...output omitted...  
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`.

chapitre 4 | Gestion des images de conteneur

```
[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
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a28866156671f0bbbb
```

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 :

- Container name : `my-nginx`
- Run as daemon : oui
- Container image : `do180/mynginx:v1.0`
- Port forward : depuis le port d'hôte 8280 vers le 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.
- 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 rétropartis 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.

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 de connaître la syntaxe Containerfile et utilisent des commandes de système d'exploitation

chapitre 5 | Création d'images de conteneur personnalisées

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-en deux.)**
 - 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-en deux.)**
- 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.

chapitre 5 | Création d'images de conteneur personnalisées

- 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 :

- La forme exécutable (à l'aide d'une matrice JSON) :

```
ENTRYPPOINT ["command", "param1", "param2"]
CMD ["param1", "param2"]
```

- La 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 remplaçable 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 :

- La forme shell :

```
ADD <_source_ >... <_destination_ >  
COPY <_source_ >... <_destination_>
```

- La 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://someserver.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 décompresse l'archive en tant que répertoire dans le dossier de *destination*. L'instruction COPY n'a pas cette fonctionnalité.



Mise en garde

Les instructions ADD et COPY copient les fichiers, en conservant les autorisations, avec root en tant que propriétaire, même si l'instruction USER est spécifiée. Red Hat vous recommande d'utiliser une instruction RUN après la copie pour changer le propriétaire et éviter les erreurs de type permission denied (autorisation refusée).

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 :

```
$ 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/].



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 /home/student/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 /home/student/D0180/labs/dockerfile-create
[student@workstation dockerfile-create]$ podman build --layers=false \
> -t do180/apache .
STEP 1: FROM ubi8/ubi:8.5
Getting image source signatures ①
Copying blob sha256:...output omitted...
71.46 MB / 71.46 MB [=====] 18s
...output omitted...
Storing signatures
STEP 2: MAINTAINER Your Name <youremail>
STEP 3: LABEL description="A custom Apache container based on UBI 8"
STEP 4: RUN yum install -y httpd &&      yum clean all
Loaded plugins: ovl, product-id, search-disabled-repos, subscription-manager
...output omitted...
STEP 5: RUN echo "Hello from Containerfile" > /var/www/html/index.html
STEP 6: EXPOSE 80
STEP 7: CMD ["httpd", "-D", "FOREGROUND"]
STEP 8: COMMIT ...output omitted... localhost/do180/apache:latest
Getting image source signatures
...output omitted...
Storing signatures
b49375fa8ee1e549dc1b72742532f01c13e0ad5b4a82bb088e5befbe59377bcf
```

- 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   8ebfe343e08c  15 seconds ago  234
MB
registry.access.redhat.com/ubi8/ubi    8.5      4199acc83c6a  6 weeks ago   213
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 ...output omitted...
fa1d1c450e88 localhost/do180/apache:latest httpd -D FOREGROU...output omitted...
```

3.3. Utilisez la commande `curl` pour vérifier que le serveur est en cours d'exécution.

```
[student@workstation dockerfile-create]$ curl 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 8.3 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 `/home/student/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
```

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. Assurez-vous que le serveur traite le fichier 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`, exéutez 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 8.3 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 `/home/student/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
```

- Utilisez votre éditeur préféré pour modifier le `Containerfile` situé dans le dossier `/home/student/D0180/labs/dockerfile-review/`.

```
[student@workstation ~]$ cd /home/student/D0180/labs/dockerfile-review/  
[student@workstation dockerfile-review]$ vim Containerfile
```

- Définissez l'image de base du fichier `Containerfile` sur `ubi8/ubi:8.5`.

```
FROM ubi8/ubi:8.5
```

- Définissez vos nom et adresse électronique avec l'instruction `MAINTAINER`.

```
MAINTAINER Your Name <youremail>
```

- Créez une variable d'environnement nommée `PORT` et configurez-la sur 8080.

```
ENV PORT 8080
```

- Installez le serveur Apache.

```
RUN yum install -y httpd && \  
      yum clean all
```

- 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/
```

- 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}
```

- Copiez tous les fichiers du dossier `src` dans Apache `DocumentRoot` sous `/var/www/html`.

chapitre 5 | Création d'images de conteneur personnalisées

```
# 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: FROM ubi8/ubi:8.5
...output omitted...
STEP 2: MAINTAINER username <username@example.com>
STEP 3: ENV PORT 8080
STEP 4: RUN yum install -y httpd &&      yum clean all
...output omitted...
STEP 5: 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/
```

chapitre 5 | Création d'images de conteneur personnalisées

```
STEP 6: USER apache
STEP 7: EXPOSE ${PORT}
STEP 8: COPY ./src/ /var/www/html/
STEP 9: CMD ["httpd", "-D", "FOREGROUND"]
STEP 10: COMMIT ...output omitted... localhost/do180/custom-apache:latest
...output omitted...
```

- 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   08fcf6d92b16  3 minutes ago
234 MB
registry.access.redhat.com/ubi8/ubi       8.3     4199acc83c6a  6 weeks ago
213 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
367823e35c4a...
```

- 3.2. Vérifiez que le conteneur est prêt et en cours d'exécution.

```
[student@workstation dockerfile-review]$ podman ps
... IMAGE           COMMAND           ... PORTS          NAMES
... do180/custom... "httpd -D ..." ... 0.0.0.0:20080->8080/tcp  containerfile
```

4. Assurez-vous que le serveur traite le fichier HTML.

- 4.1. Exécutez une commande `curl` sur `127.0.0.1:20080`

```
[student@workstation dockerfile-review]$ curl 127.0.0.1:20080
```

Vous devriez obtenir une sortie semblable à ceci :

```
<html>
<header><title>DO180 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.
- La création d'une image à partir d'un Containerfile est un processus en trois étapes :
 1. Créer un répertoire de travail
 2. Spécifiez les instructions de création dans un fichier **Containerfile**.
 3. Créez l'image à l'aide de la commande `podman build`.
- 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.

chapitre 6

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.
- Construire une application au moyen de la fonction Source-to-Image d'OpenShift Container Platform.
- Créer une application à l'aide de la console Web OpenShift.

Sections

- Description de l'architecture de Kubernetes et d'OpenShift (avec quiz)
- Création de ressources Kubernetes (et exercice guidé)
- Création de routes (et exercice guidé)
- Création d'applications avec la fonction Source-to-Image (avec exercice guidé)
- Création d'applications avec la console Web OpenShift (et exercice guidé)

Atelier

- Déploiement d'applications en conteneur dans OpenShift

Description de l'architecture de Kubernetes et d'OpenShift

Résultats

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

- Décrire l'architecture d'un cluster Kubernetes s'exécutant sur Red Hat OpenShift Container Platform (RHOCP).
- Lister les principaux types de ressources fournis par Kubernetes et RHOCP.
- Identifier les caractéristiques réseau des conteneurs, de Kubernetes et de RHOCP.
- Citer les mécanismes permettant de rendre un pod disponible en externe.

Kubernetes et OpenShift

Dans les chapitres précédents, nous avons vu que Kubernetes est un service d'orchestration qui simplifie le déploiement, la gestion et la mise à l'échelle des applications en conteneur. L'un des principaux avantages de l'utilisation de Kubernetes est qu'il utilise plusieurs nœuds pour assurer la résilience et l'évolutivité de ses applications gérées. Kubernetes forme un cluster de serveurs nœuds qui exécutent des conteneurs et sont gérés de manière centralisée par un ensemble de serveurs de plan de contrôle. Un serveur peut agir à la fois en tant que nœud de plan de contrôle et que nœud de calcul, mais ces rôles sont généralement séparés pour une meilleure stabilité.

Terminologie Kubernetes

Terme	Définition
Nœud	Serveur qui héberge des applications dans un cluster Kubernetes.
Plan de contrôle	Fournit des services de cluster de base tels que des API ou des contrôleurs.
Nœud de calcul	Ce nœud exécute des charges de travail pour le cluster. Les pods d'applications sont planifiés sur les nœuds de calcul.
Ressource	Les ressources correspondent à tout type de définition de composant géré par Kubernetes. Les ressources contiennent la configuration du composant géré (par exemple, le rôle attribué à un nœud) et l'état actuel du composant (par exemple, si le nœud est disponible).
Contrôleur	Un contrôleur est un processus Kubernetes qui surveille les ressources et effectue des modifications pour tenter de modifier l'état actuel vers l'état souhaité.
Étiquette	Une paire clé-valeur qui peut être affectée à toute ressource Kubernetes. Les sélecteurs utilisent des étiquettes pour filtrer les ressources pouvant faire l'objet d'une planification et d'autres opérations.

Terme	Définition
Namespace	Étendue pour les ressources et les processus Kubernetes, afin que les ressources portant le même nom puissent être utilisées dans différentes limites.

**Note**

Les dernières versions de Kubernetes mettent en œuvre de nombreux contrôleurs en tant qu'opérateurs. Les opérateurs sont des composants plug-in Kubernetes capables de réagir aux événements de cluster et de contrôler l'état des ressources. Les opérateurs et CoreOS Operator Framework n'entrent pas dans le cadre de ce document.

Red Hat OpenShift Container Platform est un ensemble de composants modulaires et de services créés par-dessus Red Hat CoreOS et Kubernetes. RHOCP ajoute des fonctionnalités PaaS telles que la gestion à distance, 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.

Un cluster OpenShift est un cluster Kubernetes qui peut être géré de la même façon, mais à l'aide des outils de gestion fournis par OpenShift, tels que l'interface de ligne de commande ou la console Web. Cela permet d'obtenir des workflows plus productifs et facilite l'exécution des tâches courantes.

Terminologie OpenShift

Terme	Définition
Nœud infra	Serveur de nœud contenant des services d'infrastructure tels que la surveillance, la journalisation ou le routage externe.
Console	Interface utilisateur Web fournie par le cluster RHOCP qui permet aux développeurs et aux administrateurs d'interagir avec les ressources du cluster.
Projet	Extension OpenShift des espaces de noms Kubernetes. Permet la définition du contrôle d'accès des utilisateurs (UAC) aux ressources.

Le schéma suivant illustre la pile OpenShift Container Platform :

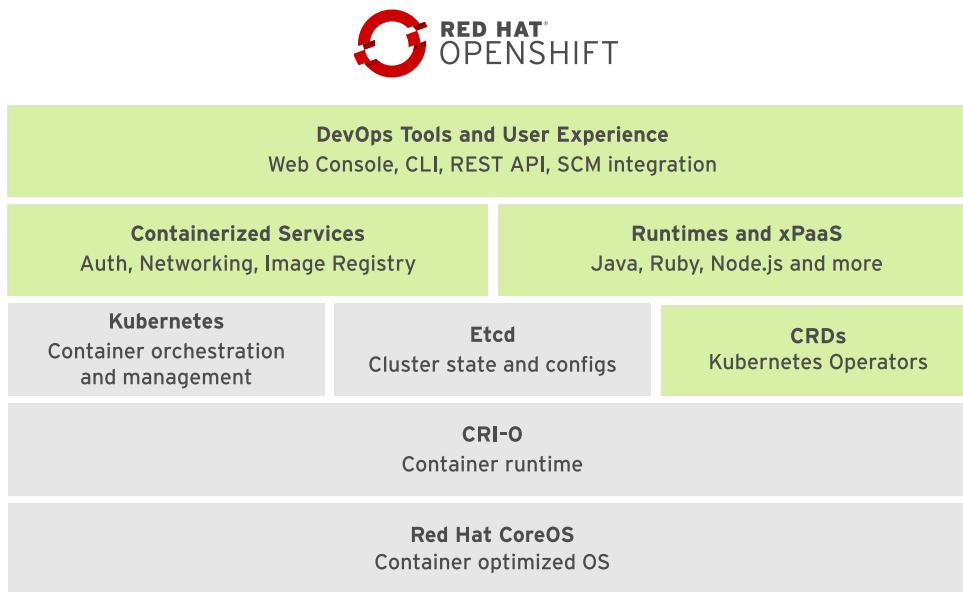


Figure 6.1: Pile de composants OpenShift

De bas en haut et de gauche à droite, cette figure présente l'infrastructure de conteneurs de base, intégrée et optimisée par Red Hat :

- Le système d'exploitation de base est Red Hat CoreOS. Red Hat CoreOS est une distribution Linux axée sur la fourniture d'un système d'exploitation immuable pour l'exécution de conteneurs.
- CRI-O est une implémentation de l'interface CRI (Container Runtime Interface) de Kubernetes permettant d'utiliser les runtimes compatibles OCI (Open Container Initiative). CRI-O permet d'utiliser n'importe quelle version d'exécution de conteneur conforme à la CRI : runc (utilisé par le service Docker), libpod (utilisé par Podman) ou rkt (de CoreOS).
- Kubernetes gère un cluster d'hôtes, physiques ou virtuels, qui exécutent des conteneurs. Il fonctionne avec des ressources qui décrivent les applications multiconteneurs composées de ressources multiples et la manière dont elles se connectent entre elles.
- Etcd est un magasin de clés/valeurs utilisé par Kubernetes pour stocker des informations de configuration et d'état sur les conteneurs et autres ressources se trouvant à l'intérieur du cluster Kubernetes.
- Les définitions de ressources personnalisées (CRD) sont des types de ressources stockées dans Etcd et gérées par Kubernetes. Ces types de ressources forment l'état et la configuration de toutes les ressources gérées par OpenShift.
- Les services en conteneur remplissent de nombreuses fonctions d'infrastructure PaaS, telles que la gestion réseau et l'autorisation. RHOPC utilise l'infrastructure de conteneurs de base de Kubernetes et l'exécutable de conteneur sous-jacent pour la plupart des fonctions internes. Autrement dit, la plupart des services internes de RHOPC sont exécutés en tant que conteneurs orchestrés par Kubernetes.
- Les runtimes et xPaaS sont des images de conteneur de base prêtes à être utilisées par les développeurs. Chacune est préconfigurée avec un langage d'exécution ou une base de données spécifique. L'offre xPaaS est un ensemble d'images de base pour les produits Middleware Red Hat, tels que JBoss EAP et ActiveMQ. RHOR (Red Hat OpenShift Application Runtimes) est un ensemble de runtimes optimisé pour les applications natives cloud dans OpenShift.

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

Les runtimes d'applications disponibles sont Red Hat JBoss EAP, OpenJDK, Thorntail, Eclipse Vert.x, SpringBoot et Node.js.

- RHOCP fournit des outils de gestion de l'interface utilisateur web et de la CLI pour gérer les applications et les services RHOCP. Les outils de l'interface utilisateur web et CLI OpenShift sont élaborés à partir des API REST, qui peuvent être utilisées par des outils externes tels que les IDE et les plateformes CI.

Cette illustration d'architecture OpenShift et Kubernetes permet de comprendre comment les composants de l'infrastructure fonctionnent ensemble.

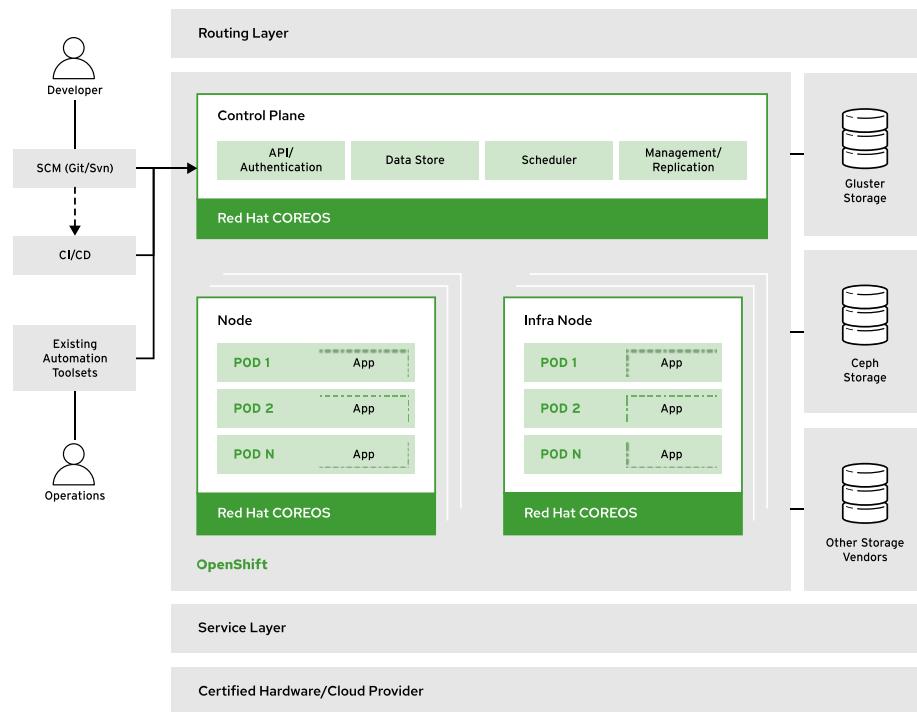


Figure 6.2: Architecture OpenShift et Kubernetes

Nouvelles fonctionnalités dans RHOCP 4

RHOCP 4 est un changement massif par rapport aux versions précédentes. En plus de conserver une compatibilité ascendante avec les versions précédentes, il inclut de nouvelles fonctionnalités, telles que :

- CoreOS en tant que système d'exploitation obligatoire pour tous les nœuds, offrant une infrastructure immuable optimisée pour les conteneurs.
- Un tout nouveau programme d'installation de cluster qui guide le processus d'installation et de mise à jour.
- Une plateforme autogérée, capable d'appliquer automatiquement les mises à jour et les restaurations du cluster sans interruption.
- Une gestion repensée du cycle de vie des applications.
- Un kit de développement logiciel (SDK) de l'opérateur pour créer, tester et empaqueter des opérateurs.

Description des types de ressources Kubernetes

Kubernetes comprend six types de ressources principaux qui peuvent être créés et configurés à l'aide d'un fichier YAML ou JSON, ou au moyen d'outils de gestion OpenShift :

Pods (po)

Les pods représentent un ensemble de conteneurs qui partagent des ressources, telles que des adresses IP et des volumes de stockage persistants. Il s'agit de l'unité de travail de base pour Kubernetes.

Services (svc)

Ils définissent une combinaison IP/port unique qui fournit l'accès à un pool de pods. Par défaut, les services connectent des clients à des pods à tour de rôle.

Contrôleurs de réPLICATION (rc)

Une ressource Kubernetes qui définit le mode de réPLICATION des pods (évolutivité horizontale) sur différents nœuds. Les contrôleurs de réPLICATION sont un service Kubernetes de base destiné à fournir une haute disponibilité aux pods et conteneurs.

Volumes persistants (pv)

Définit les zones de stockage à utiliser par les pods Kubernetes.

Revendications de volume persistant (pvc)

Représenter une demande de stockage adressée par un pod. Les PVC relient un PV à un pod afin que ses conteneurs puissent l'utiliser, généralement en montant le stockage dans le système de fichiers du conteneur.

ConfigMaps (cm) et Secrets

Contient un ensemble de clés et de valeurs pouvant être utilisées par d'autres ressources. Les ConfigMaps et Secrets sont généralement utilisés pour centraliser les valeurs de configuration utilisées par plusieurs ressources. Les Secrets diffèrent des mises en correspondance ConfigMaps en ce sens que les valeurs des Secrets sont toujours codées (et non pas chiffrées) et leur accès est limité à un nombre moins élevé d'utilisateurs autorisés.



Note

Bien que les pods Kubernetes puissent être créés de manière autonome, ils le sont généralement par des ressources de niveau supérieur, telles que des contrôleurs de réPLICATION.

Types de ressources OpenShift

Les principaux types de ressources ajoutés par OpenShift Container Platform à Kubernetes sont les suivants :

Deployment et Deployment config (dc)

OpenShift 4.5 a introduit le concept de ressource Deployment pour remplacer DeploymentConfig en tant que configuration par défaut pour les pods. Tous deux sont la représentation d'un ensemble de conteneurs inclus dans un pod et des stratégies de déploiement à utiliser. Il contient la configuration à appliquer à tous les conteneurs de chaque réplique de pod, tels que l'image de base, les balises, les définitions de stockage et les commandes à exécuter au démarrage des conteneurs.

L'objet Deployment fait office de version améliorée de l'objet DeploymentConfig. Voici quelques remplacements de fonctionnalités entre les deux objets :

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

- La restauration automatique n'est plus prise en charge par les objets de déploiement.
- Chaque modification du modèle de pod utilisé par les objets de déploiement déclenche automatiquement un nouveau déploiement.
- Les scripts automatiques de cycle de vie ne sont plus pris en charge par les objets de déploiement.
- Le processus de déploiement d'un objet de déploiement peut être mis en pause à tout moment sans affecter le processus du programme de déploiement.
- Un objet de déploiement peut avoir autant de ensembles de répliques actifs que l'utilisateur le souhaite, en réduisant les anciennes répliques après un certain temps. En revanche, l'objet deploymentconfig ne peut avoir que deux ensembles de réPLICATION actifs en même temps.

Même si les objets Deployment sont destinés à faire office de remplacement par défaut des objets DeploymentConfig, dans OpenShift 4.10, les utilisateurs peuvent toujours les utiliser s'ils ont besoin d'une fonctionnalité spécifique fournie par ces objets. Dans ce cas, il est nécessaire de spécifier le type d'objet lors de la création d'une application en spécifiant l'indicateur --as-deployment-config.

Configuration de compilation (bc)

Définit un processus à exécuter dans le projet OpenShift. Utilisée par la fonction Source-to-Image (S2I) d'OpenShift pour compiler une image de conteneur à partir d'un code source d'application stocké dans un référentiel Git. Une bc collabore avec une dc pour fournir des workflows d'intégration et de déploiement continu de base, mais extensibles.

Routes

Les routes représentent un nom d'hôte DNS reconnu par le routeur OpenShift comme un point d'entrée pour les applications et les microservices.



Note

Pour obtenir la liste de toutes les ressources disponibles dans un cluster RHOCP et leurs abréviations, utilisez les commandes oc api-resources ou kubectl api-resources.

Bien que les contrôleurs de réPLICATION Kubernetes puissent être créés de manière autonome dans OpenShift, ils le sont généralement par des ressources de niveau supérieur, telles que des contrôleurs de déploiement.

Réseau

Chaque conteneur déployé dans un cluster Kubernetes possède une adresse IP affectée à partir d'un réseau interne accessible uniquement depuis le nœud qui l'exécute. Compte tenu du caractère éphémère du conteneur, les adresses IP sont constamment affectées et libérées.

Kubernetes fournit un réseau SDN (Software-Defined Network) (SDN) qui génère les réseaux de conteneurs internes à partir de plusieurs nœuds et permet aux conteneurs de n'importe quel pod, à l'intérieur d'un hôte, d'accéder aux pods d'autres hôtes. L'accès au réseau SDN n'est possible qu'à partir du même réseau Kubernetes.

Les conteneurs qui se trouvent à l'intérieur des pods Kubernetes ne doivent pas se connecter directement à l'adresse IP dynamique de leurs homologues. Les services résolvent ce problème en liant des adresses IP plus stables à partir du SDN vers les pods. Si les pods sont redémarrés, répliqués ou replanifiés sur des nœuds différents, les services sont mis à jour, offrant ainsi une évolutivité et une tolérance aux pannes.

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

L'accès externe aux conteneurs est plus compliqué. Les services Kubernetes peuvent être définis en tant que type de service `NodePort`. Avec ce type de service, Kubernetes alloue un port réseau à partir d'une plage prédéfinie dans chacun des nœuds du cluster et le met en proxy dans votre service. Malheureusement, cette méthode n'offre pas une évolutivité idéale.

Avec OpenShift, l'accès externe aux conteneurs est à la fois évolutif et plus simple grâce à la définition de ressources de route. Une route définit des ports et des noms DNS externes pour un service. Un routeur (contrôleur d'entrée) transmet les requêtes HTTP et TLS aux adresses de service à l'intérieur du SDN Kubernetes. La seule exigence concerne la mise en correspondance des noms DNS souhaités avec les adresses IP des nœuds des routeurs RHOP.

**Références****Site Web de documentation de Kubernetes**

<https://kubernetes.io/docs/>

Site Web de documentation OpenShift

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

Présentation des opérateurs

<https://docs.openshift.com/container-platform/4.10/operators/understanding/olm-what-operators-are.html>

► Quiz

Description de Kubernetes et d'OpenShift

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

► 1. **Parmi les propositions suivantes concernant l'architecture Kubernetes, lesquelles sont correctes ? (Choisissez-en deux.)**

- a. Les nœuds Kubernetes peuvent être gérés sans plan de contrôle.
- b. Les plans de contrôle Kubernetes gèrent la mise à l'échelle des pods.
- c. Les plans de contrôle Kubernetes planifient des pods sur des nœuds spécifiques.
- d. Les outils Kubernetes ne peuvent pas être utilisés pour gérer des ressources dans un cluster OpenShift.
- e. Les conteneurs créés à partir de pods Kubernetes ne peuvent pas être gérés à l'aide d'outils autonomes tels que Podman.

► 2. **Parmi les propositions suivantes concernant les types de ressources Kubernetes et OpenShift, lesquelles sont correctes ? (Choisissez-en deux.)**

- a. Un pod est chargé de déployer son propre espace de stockage persistant.
- b. Tous les pods générés à partir du même contrôleur de réPLICATION doivent s'exécuter dans le même mode.
- c. Un volume persistant définit des zones de stockage disponibles pour les pods par le biais d'une réclamation de volume persistant.
- d. Un contrôleur de réPLICATION est chargé de contrôler et de maintenir le nombre de pods d'une application donnée.

► 3. **Parmi les propositions suivantes concernant les réseaux Kubernetes et OpenShift, lesquelles sont vraies ? (Choisissez-en deux.)**

- a. Un service Kubernetes peut fournir une adresse IP pour accéder à un ensemble de pods.
- b. Kubernetes est chargé de fournir un nom de domaine complet pour un pod.
- c. Un contrôleur de réPLICATION est chargé de l'acheminement des requêtes externes vers les pods.
- d. Une route est chargée de fournir des noms DNS pour un accès externe.

- 4. Parmi les propositions suivantes concernant le stockage persistant dans OpenShift et Kubernetes, laquelle est correcte ?
- a. Une PVC représente une zone de stockage qu'un pod peut utiliser pour stocker des données ; elle est déployée par le développeur de l'application.
 - b. Une PVC représente une zone de stockage qui peut être demandée par un pod pour stocker des données, mais elle est déployée par l'administrateur du cluster.
 - c. Une PVC représente la quantité de mémoire qui peut être allouée sur un nœud, de sorte qu'un développeur puisse spécifier la quantité nécessaire à l'exécution de son application.
 - d. Une PVC représente le nombre d'unités de traitement pouvant être allouées sur un pod d'application, sous réserve d'une limite gérée par l'administrateur du cluster.
- 5. Parmi les propositions suivantes concernant les ajouts d'OpenShift dans Kubernetes, laquelle est correcte ?
- a. OpenShift ajoute des fonctionnalités pour simplifier la configuration Kubernetes de nombreux cas d'utilisation réels.
 - b. Les images de conteneur créées pour OpenShift ne peuvent pas être utilisées avec Kubernetes.
 - c. Red Hat conserve les versions scindées de Kubernetes à l'intérieur du produit RHOC.
 - d. Des outils externes sont nécessaires pour procéder à l'intégration continue et au déploiement continu avec RHOC.

► Solution

Description de Kubernetes et d'OpenShift

Répondez aux questions suivantes en sélectionnant un ou plusieurs éléments :

- 1. **Parmi les propositions suivantes concernant l'architecture Kubernetes, lesquelles sont correctes ? (Choisissez-en deux.)**
- a. Les nœuds Kubernetes peuvent être gérés sans plan de contrôle.
 - b. Les plans de contrôle Kubernetes gèrent la mise à l'échelle des pods.
 - c. Les plans de contrôle Kubernetes planifient des pods sur des nœuds spécifiques.
 - d. Les outils Kubernetes ne peuvent pas être utilisés pour gérer des ressources dans un cluster OpenShift.
 - e. Les conteneurs créés à partir de pods Kubernetes ne peuvent pas être gérés à l'aide d'outils autonomes tels que Podman.
- 2. **Parmi les propositions suivantes concernant les types de ressources Kubernetes et OpenShift, lesquelles sont correctes ? (Choisissez-en deux.)**
- a. Un pod est chargé de déployer son propre espace de stockage persistant.
 - b. Tous les pods générés à partir du même contrôleur de réplication doivent s'exécuter dans le même mode.
 - c. Un volume persistant définit des zones de stockage disponibles pour les pods par le biais d'une réclamation de volume persistant.
 - d. Un contrôleur de réplication est chargé de contrôler et de maintenir le nombre de pods d'une application donnée.
- 3. **Parmi les propositions suivantes concernant les réseaux Kubernetes et OpenShift, lesquelles sont vraies ? (Choisissez-en deux.)**
- a. Un service Kubernetes peut fournir une adresse IP pour accéder à un ensemble de pods.
 - b. Kubernetes est chargé de fournir un nom de domaine complet pour un pod.
 - c. Un contrôleur de réplication est chargé de l'acheminement des requêtes externes vers les pods.
 - d. Une route est chargée de fournir des noms DNS pour un accès externe.

- 4. **Parmi les propositions suivantes concernant le stockage persistant dans OpenShift et Kubernetes, laquelle est correcte ?**
- a. Une PVC représente une zone de stockage qu'un pod peut utiliser pour stocker des données ; elle est déployée par le développeur de l'application.
 - b. Une PVC représente une zone de stockage qui peut être demandée par un pod pour stocker des données, mais elle est déployée par l'administrateur du cluster.
 - c. Une PVC représente la quantité de mémoire qui peut être allouée sur un nœud, de sorte qu'un développeur puisse spécifier la quantité nécessaire à l'exécution de son application.
 - d. Une PVC représente le nombre d'unités de traitement pouvant être allouées sur un pod d'application, sous réserve d'une limite gérée par l'administrateur du cluster.
- 5. **Parmi les propositions suivantes concernant les ajouts d'OpenShift dans Kubernetes, laquelle est correcte ?**
- a. OpenShift ajoute des fonctionnalités pour simplifier la configuration Kubernetes de nombreux cas d'utilisation réels.
 - b. Les images de conteneur créées pour OpenShift ne peuvent pas être utilisées avec Kubernetes.
 - c. Red Hat conserve les versions scindées de Kubernetes à l'intérieur du produit RHOC.
 - d. Des outils externes sont nécessaires pour procéder à l'intégration continue et au déploiement continu avec RHOC.

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 :

```
$> 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 :

```
$> oc login <clusterUrl>
```

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: Pod❶
metadata:
  name: wildfly❷
  labels:
    name: wildfly❸
spec:
  containers:
    - resources:
        limits:
          cpu: 0.5
      image: do276/todojee❹
      name: wildfly
      ports:
        - containerPort: 8080❺
          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 :

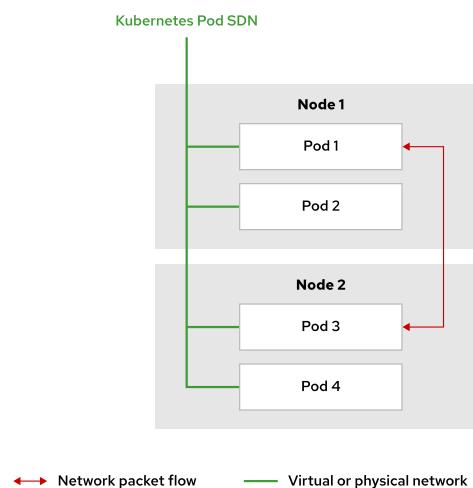


Figure 6.3: Mise en réseau Kubernetes de base

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 :

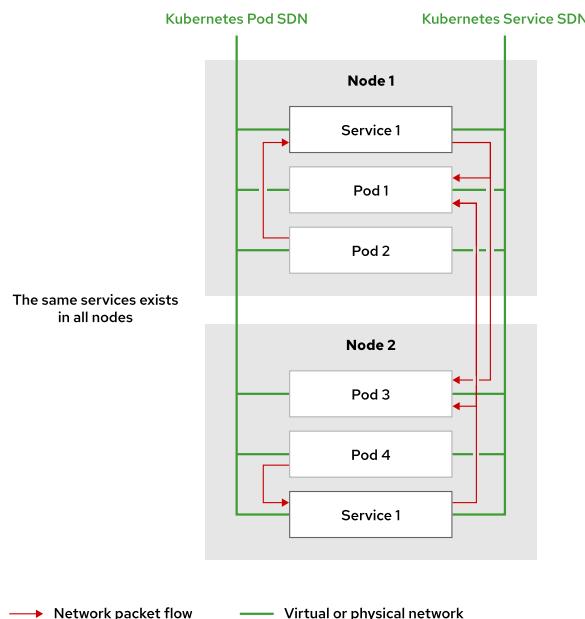


Figure 6.4: Réseau de services Kubernetes

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 du DNS : les lettres sont mises en majuscule 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
```

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

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 6.5 illustre comment les services NodePort permettent un accès externe aux services Kubernetes. Les routes OpenShift sont traitées plus en détail ultérieurement dans ce cours.

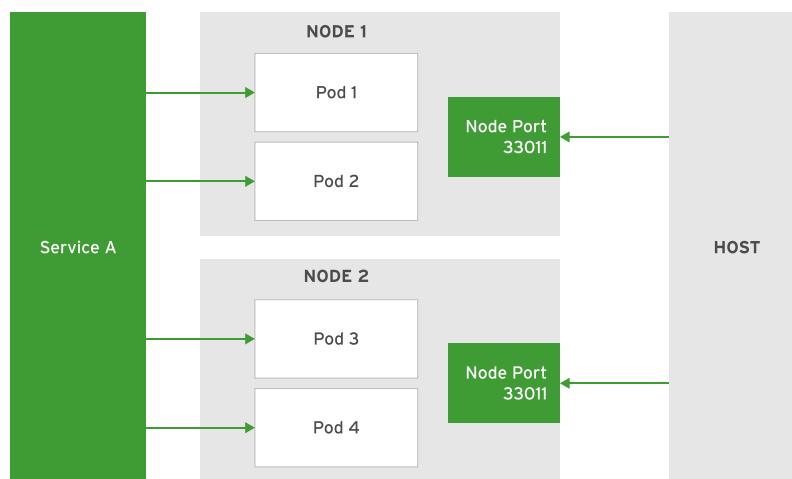


Figure 6.5: Méthode alternative pour l'accès externe à un service Kubernetes

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 :

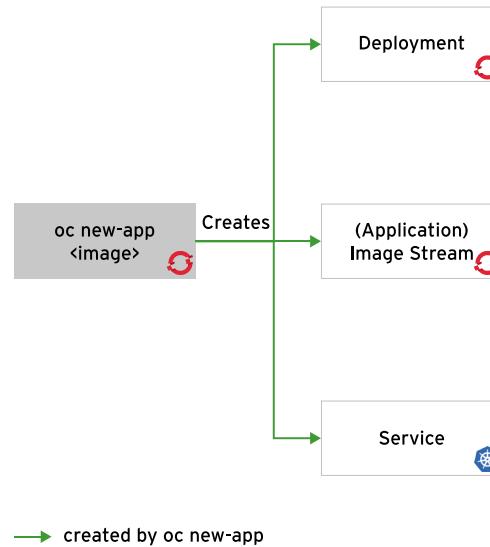


Figure 6.6: Ressources créées pour une nouvelle application

La commande suivante crée une application basée sur une image à partir d'un registre d'image Docker privé :

```
oc new-app --docker-image=myregistry.com/mycompany/myapp --name=myapp
```

La commande suivante crée une application à partir d'un code source stocké dans un référentiel Git :

```
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 :

```
$ oc new-app \
> --template=mysql-persistent \
> -p MYSQL_USER=user1 -p MYSQL_PASSWORD=mypassword -p MYSQL_DATABASE=testdb \
> -p MYSQL_ROOT_PASSWORD=rootpassword -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          
```

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" ①
  volumes:
    - name: "pvol" ②
      persistentVolumeClaim:
        claimName: "myapp" ③

```

- ① Cette section déclare que le volume `pvol` est monté sur `/var/www/html` dans le système de fichiers du conteneur.
- ② Cette section définit le volume `pvol`.
- ③ 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
<code>nginx-1-5r583</code>	1/1	Running	0	1h
<code>myapp-1-l44m7</code>	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	
<code>is/nginx</code>	<code>172.30.1.1:5000/basic-kubernetes/nginx</code>	<code>latest</code>	About an hour ago	
NAME	REVISION	DESIRED	CURRENT	TRIGGERED BY
<code>dc/nginx</code>	1	1	1	<code>config,image(nginx:latest)</code>
NAME	DESIRED	CURRENT	READY	AGE
<code>rc/nginx-1</code>	1	1	1	1h
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
<code>svc/nginx</code>	<code>172.30.72.75</code>	<code><none></code>	<code>80/TCP, 443/TCP</code>	1h
NAME	READY	STATUS	RESTARTS	AGE
<code>po/nginx-1-ypp8t</code>	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` fait office de sélecteur pour extraire uniquement les objets disposant d'une étiquette correspondante :

```
$ 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           ...
```



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

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

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-1-5vfn4 1/1     Running   0          109s
```

**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...
```

- 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 -pmypa5 --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.

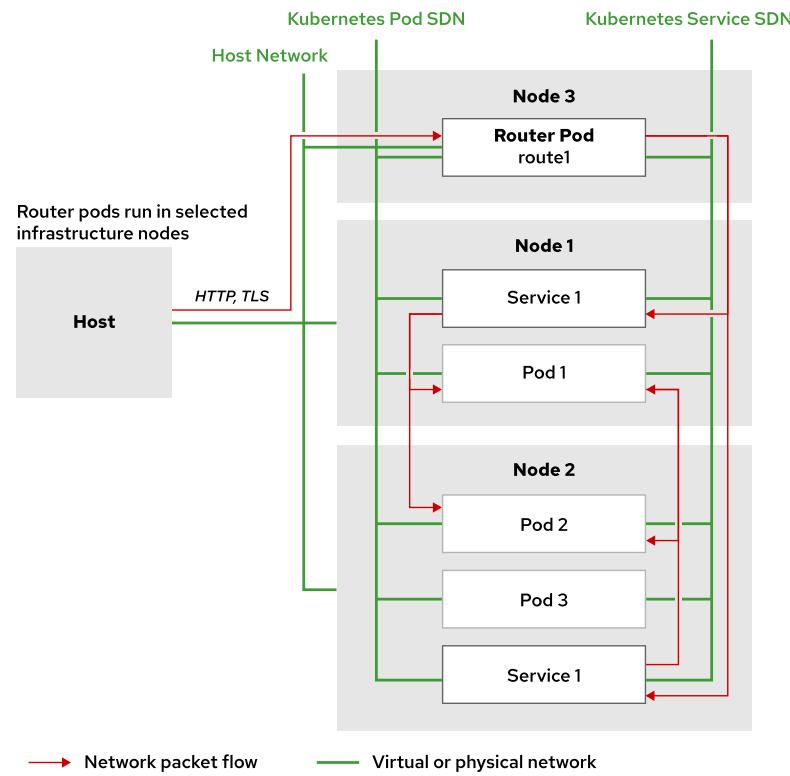


Figure 6.7: Routes OpenShift et services Kubernetes

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 :

```
$ 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 :

```
$ 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 :

```
$ oc describe pod router-default-746b5cfb65-f6sdm
Name:           router-default-746b5cfb65-f6sdm
Namespace:      openshift-ingress
...output omitted...
Containers:
  router:
    ...output omitted...
    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
...output omitted...
```

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/

► 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 \
> --docker-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:     extrdp-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:     extrdp-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-extrdp-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:           extrdp-xyz
Namespace:      extrdp-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: extrdp-xyz-extrdp-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`, exécutez 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 :

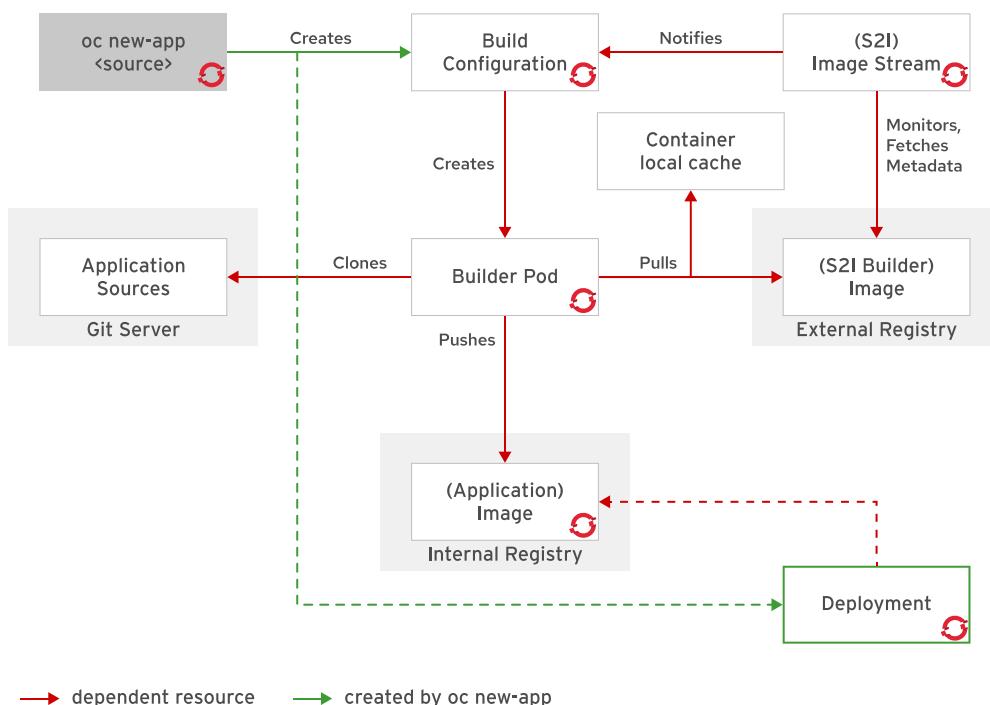


Figure 6.8: Déploiement et ressources dépendantes

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 :

```
$ 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.

chapitre 6 | Déploiement d'applications en conteneur dans 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 :

```
$ 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.

```
$ 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 :

```
$ 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 :

```
$ 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 :

```
$ 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 Gemfile config.ru</code>
Outils	<code>pom.xml</code>
Node.js	<code>app.json package.json</code>
PHP	<code>index.php composer.json</code>
Python	<code>requirements.txt config.py</code>
Perl	<code>index.pl 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 :

```
$ 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": "3q8K8JNDoRzhjoz1KgMz"  
                }  
            },  
            {  
                "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...
```

- ①** Définissez un type de ressource **BuildConfig**.
- ②** Nommez la ressource **BuildConfig** **myapp**.
- ③** Définissez l'adresse du référentiel Git du code source.
- ④** Définissez la stratégie d'utilisation de S2I.
- ⑤** Définissez l'image d'outil de compilation en tant que flux d'images **php:7.3**.
- ⑥** 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\"}, \"fieldPath\": \"spec.template.spec.containers[?(@.name==\\\\\"myapp\\\\\").image\"]\"}], ③
                \"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
    }
],
"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 :

```
$ 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 :

```
$ 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` :

```
$ oc get buildconfig
NAME          TYPE      FROM      LATEST
myapp        Source   Git       1
```

```
$ 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

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 devez pouvoir réaliser les tâches 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 openshift-s2i]$ 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 `/home/student/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
```

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

```
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}
```

3.7. Testez l'application en envoyant une requête HTTP GET à l'URL que vous avez obtenue lors de l'étape précédente :

```
[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
```

- 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 sert le nouveau contenu :

```
[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.

Création d'applications avec la console Web OpenShift

Résultats

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

- Créer une application à l'aide de la console Web OpenShift.
- Gérer et contrôler le cycle de build d'une application.
- Examiner les ressources d'une application.

Accès à la console Web OpenShift

La console Web OpenShift permet aux utilisateurs d'exécuter un grand nombre de tâches identiques à la ligne de commande OpenShift. Vous pouvez créer des projets, ajouter des applications à des projets, afficher les ressources de l'application et manipuler les configurations d'application selon les besoins. La console Web OpenShift s'exécute sous la forme d'un ou de plusieurs pods, chaque pod s'exécutant sur un plan de contrôle.

Bien que le client de ligne de commande soit plus polyvalent que la console Web, il se peut que la représentation visuelle des concepts dans OpenShift soit utile.

La console Web s'exécute dans un navigateur Web.

L'URL par défaut a le format `https://console-openshift-console.{wildcard DNS domain for the RHOC cluster}`. Par défaut, OpenShift génère un certificat auto-signé pour la console Web. Vous devez faire confiance à ce certificat afin d'obtenir l'accès.

La console Web utilise une API REST pour communiquer avec le cluster OpenShift. Par défaut, le point d'accès de l'API REST est accessible avec un nom DNS et un certificat auto-signé différent. Vous devez également faire confiance à ce certificat pour le point d'accès de l'API REST.

Une fois que vous avez approuvé les deux certificats OpenShift, la console requiert une authentification pour continuer.

Gestion de projets

En cas de connexion réussie, la page **Home > Projects** affiche une liste des projets auxquels vous pouvez accéder. À partir de cette page, vous pouvez créer, modifier ou supprimer un projet.

Si vous êtes autorisé à afficher les métriques du cluster, vous êtes alors redirigé vers la page **Home > Overview**. Cette page affiche des informations générales et des mesures concernant le cluster. L'option de menu **> Overview** est masquée pour les utilisateurs qui ne sont pas autorisés à afficher les métriques du cluster.

Name	Display Name	Status	Requester
todo-app	No display name	Active	your-user
hello-world	No display name	Active	your-user

Figure 6.9: Page d'accueil de la console Web OpenShift

L'icône des points de suspension à la fin de chaque ligne fournit un menu avec les actions du projet. Sélectionnez l'entrée appropriée pour modifier ou supprimer le projet.

Si vous cliquez sur un lien de projet dans cet affichage, vous êtes redirigé vers la page **Project Status** qui affiche toutes les applications créées dans cet espace de projet.

Navigation dans la console Web

Un menu de navigation est situé sur le côté gauche de la console Web. Ce menu comprend deux perspectives : **Administrator** et **Developer**. Chaque élément du menu se développe pour donner accès à un ensemble de fonctions de gestion connexes. Lorsque la perspective **Administrator** est sélectionnée, ces éléments sont les suivants :

Home

Le menu Home permet aux utilisateurs d'accéder rapidement à des projets et à des ressources. À partir de ce menu, vous pouvez parcourir et gérer des projets, rechercher ou explorer des ressources de cluster, et inspecter des événements de cluster.

Opérateurs

OperatorHub est un catalogue qui vous permet de découvrir, de rechercher et d'installer des opérateurs dans le cluster. Après avoir installé un opérateur, vous pouvez utiliser l'option **Installed Operators** pour gérer l'opérateur ou rechercher d'autres opérateurs installés.



Note

Les dernières versions de Kubernetes mettent en œuvre de nombreux contrôleurs en tant qu'opérateurs. Les opérateurs sont des composants plug-in Kubernetes capables de réagir aux événements de cluster et de contrôler l'état des ressources. Les opérateurs et CoreOS Operator Framework n'entrent pas dans le cadre de ce document.

Charges de travail

Ces options permettent de gérer plusieurs types de ressources Kubernetes et OpenShift, telles que les pods et les déploiements. D'autres options de déploiement avancées accessibles à partir de ce menu, telles que les mappages de configuration, les secrets et les tâches cron, sortent du cadre du cours.

Réseau

Ce menu contient des options permettant de gérer les ressources OpenShift qui affectent l'accès aux applications, telles que les services et les routes, pour un projet. D'autres options permettant de configurer une politique réseau ou une entrée OpenShift sont disponibles, mais ces rubriques sortent du cadre de ce cours.

Stockage

Ce menu contient des options permettant de configurer le stockage persistant pour les applications de projet. En particulier, les volumes persistants et les revendications de volume persistant pour un projet sont gérés à partir du menu **Storage**.

Builds

L'option **Build Configs** affiche une liste des configurations de compilation du projet. Cliquez sur un lien de configuration de compilation dans cet affichage pour accéder à une page de présentation de la configuration de compilation spécifiée. À partir de cette page, vous pouvez afficher et modifier la configuration de compilation de l'application.

L'option **Builds** fournit une liste des processus de compilation récents pour les images de conteneur d'applications dans le projet. Cliquez sur le lien d'une build particulière pour accéder aux journaux de compilation pour ce processus de compilation particulier.

L'option **Image Streams** fournit une liste des flux d'images définis dans le projet. Cliquez sur une entrée de flux d'images dans cette liste pour accéder à une page d'aperçu permettant d'afficher et de gérer ce flux d'images.

Compute

Fournit des options pour accéder aux alertes et les gérer pour le cluster OpenShift. À partir de cette page, vous pouvez également configurer la mise à l'échelle automatique et les bilans de santé des nœuds.

Gestion des utilisateurs

À partir de cette option, vous pouvez configurer l'authentification et l'autorisation à l'aide d'utilisateurs, de groupes, de rôles, de liaisons de rôle et de comptes de service.

Administration

Fournit des options permettant de gérer les paramètres de cluster et de projet, tels que les quotas de ressources et les contrôles d'accès basées sur les rôles. Les fonctions dans la section **Administration** sortent du cadre de ce cours.

Création d'applications

À partir de la perspective **Developer**, utilisez **+Add** pour sélectionner un moyen de créer une application dans un projet OpenShift. Vous pouvez ajouter une application à partir de **Developer Catalog**, en utilisant l'option **All services**, qui offre une sélection de modèles S2I (Source-to-Image), d'images de compilateur et de graphiques Helm pour créer des applications spécifiques à la technologie. Sélectionnez un modèle souhaité et fournissez les informations nécessaires pour déployer la nouvelle application.

Vous n'êtes pas limité au déploiement d'une application à partir de son code source uniquement. Vous pouvez également déployer une application en utilisant :

- Une image de conteneur hébergée sur un registre de conteneurs distant.

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

- Un fichier YAML qui spécifie les ressources Kubernetes et OpenShift à créer.
- Une image de compilateur à l'aide du code source ou d'un Containerfile de votre propre référentiel Git.

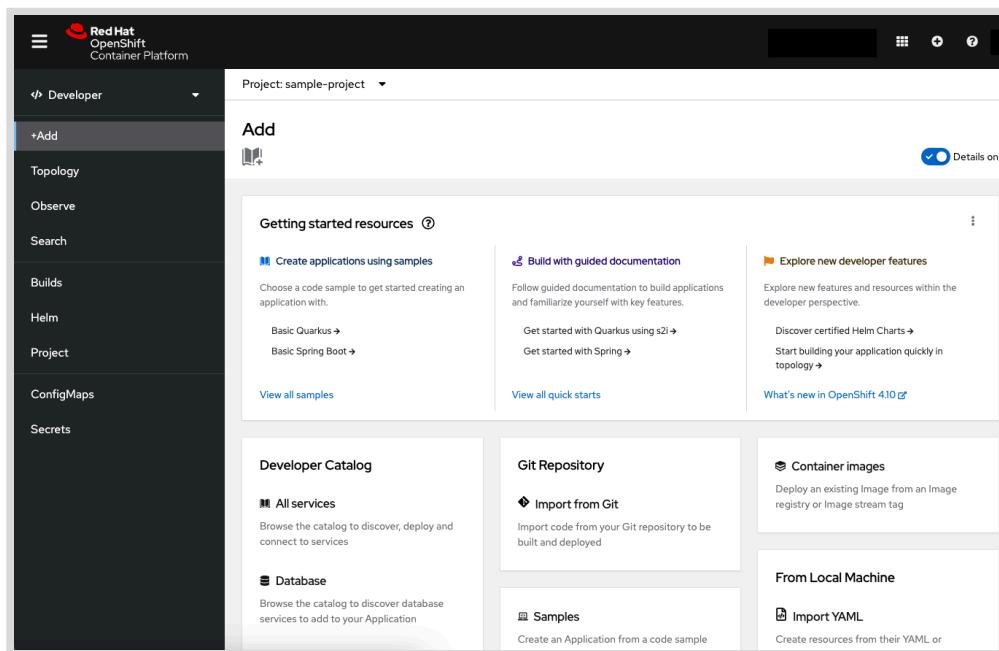


Figure 6.10: Page OpenShift Developer Catalog

Pour créer une application à l'aide de l'une de ces méthodes, sélectionnez l'option appropriée dans la page **Add**. Utilisez l'option **Container images** pour déployer une image de conteneur existante. Utilisez l'option **Import YAML** pour créer les ressources spécifiées dans un fichier YAML. Utilisez l'option **Import from Git** pour déployer votre code source à l'aide d'une image de compilateur ou votre Containerfile. Les options **Database**, **Operator Backed** et **Helm Chart** sont des raccourcis vers le catalogue.

Gestion des compilations d'application

À partir de la perspective **Administrator**, cliquez sur l'option **Build Configs** du menu **Builds** après avoir ajouté une application Source-to-Image à un projet. La nouvelle configuration de compilation est accessible à partir de cet affichage :

Figure 6.11: Page de configuration de compilation OpenShift

Cliquez sur une configuration de compilation dans la liste pour afficher une page de présentation pour la configuration de compilation choisie. À partir de la page de présentation, vous pouvez :

- Afficher les paramètres de configuration de la compilation, tels que l'URL du référentiel Git du code source.
- Afficher et modifier les variables d'environnement définies dans le conteneur de compilation lors d'un processus de création d'application.
- Afficher une liste des versions récentes de l'application, puis cliquer sur une version sélectionnée pour accéder aux journaux à partir du processus de compilation.

Gestion des applications déployées

Le menu Workloads donne accès aux déploiements dans le projet :

Figure 6.12: Menu OpenShift Workloads

Sous les charges de travail, vous trouverez de nombreuses constructions abordées dans le cours, notamment les pods, Deployments et Config Maps.

Cliquez sur une entrée de déploiements dans la liste pour afficher une page de présentation pour la sélection. À partir de la page de présentation, vous pouvez :

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

- Afficher les paramètres de déploiement, tels que les spécifications d'une image de conteneur d'application.
- Modifier le nombre souhaité de pods d'application pour redimensionner manuellement l'application.
- Afficher et modifier les variables d'environnement définies dans le conteneur d'applications déployées.
- Afficher une liste de pods d'application et cliquer sur un pod sélectionné pour accéder aux journaux de ce pod.

Autres fonctionnalités de la console Web

La console Web vous permet d'effectuer les opérations suivantes :

- Gérer des ressources telles que des quotas de projet, des adhésions, des secrets, ainsi que d'autres ressources avancées.
- Créer des revendications de volume persistant (PVC).
- Contrôler des builds, des déploiements, des pods et des événements système.
- Créer des pipelines de déploiement et d'intégration continu.

L'utilisation détaillée des fonctions ci-dessus sort du cadre de ce cours.

► Exercice guidé

Création d'une application à l'aide de la console Web

Dans cet exercice, vous allez créer, compiler et déployer une application sur un cluster OpenShift à l'aide de la console Web OpenShift.

Résultats

Vous devez pouvoir créer, compiler et déployer une application sur un cluster OpenShift à l'aide de la console 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.

Récupérez les fichiers de l'atelier en exécutant le script d'atelier :

```
[student@workstation ~]$ lab openshift-webconsole start
```

Le script de l'atelier vérifie que le cluster OpenShift est en cours d'exécution.

Instructions

- ▶ 1. Examinez le code source PHP de l'exemple d'application, et créez et envoyez par push une nouvelle branche nommée **console** 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 console
Switched to a new branch 'console'
[student@workstation D0180-apps]$ git push -u origin console
...output omitted...
 * [new branch]      console -> console
Branch 'console' set up to track remote branch 'console' 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 `/home/student/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. Ouvrez un navigateur Web et accédez à `https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}` pour accéder à la console Web OpenShift. Connectez-vous et créez un projet nommé `youruser-console`.
- 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 ~]$ source /usr/local/etc/ocp4.config
```

- 2.2. Récupérez la valeur du domaine générique spécifique à votre cluster à l'aide de `$RHT_OCP4_WILDCARD_DOMAIN`.

```
[student@workstation ~]$ echo $RHT_OCP4_WILDCARD_DOMAIN  
apps.cluster.lab.example.com
```

- 2.3. Ouvrez le navigateur Firefox et accédez à `https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}` pour accéder à la console Web OpenShift. Connectez-vous à la console OpenShift à l'aide de vos informations d'identification.
- 2.4. Create a new project named `youruser-console`. Vous pouvez saisir les valeurs que vous préférez dans les autres champs.

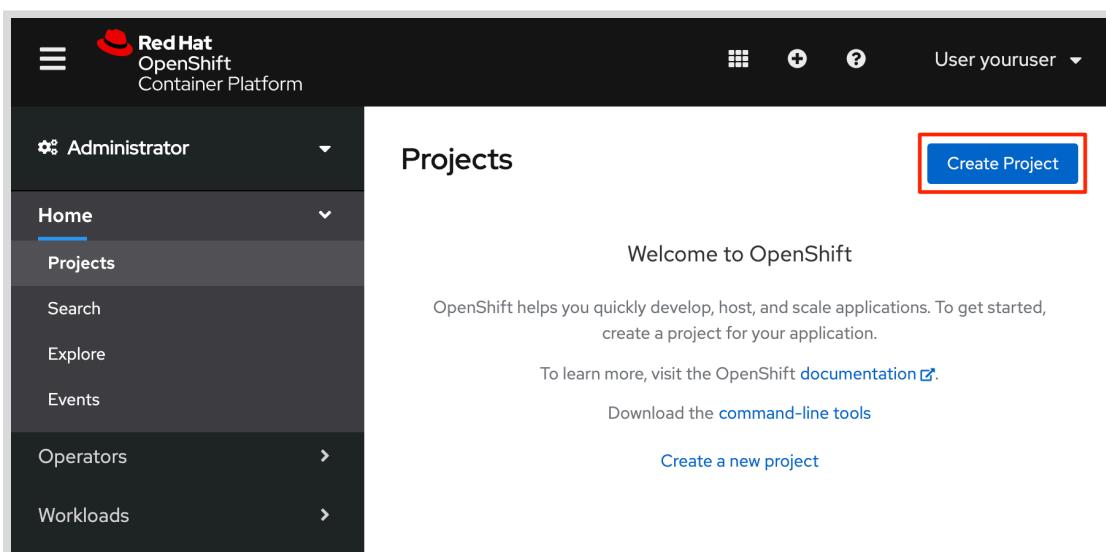


Figure 6.13: Crédit d'un projet

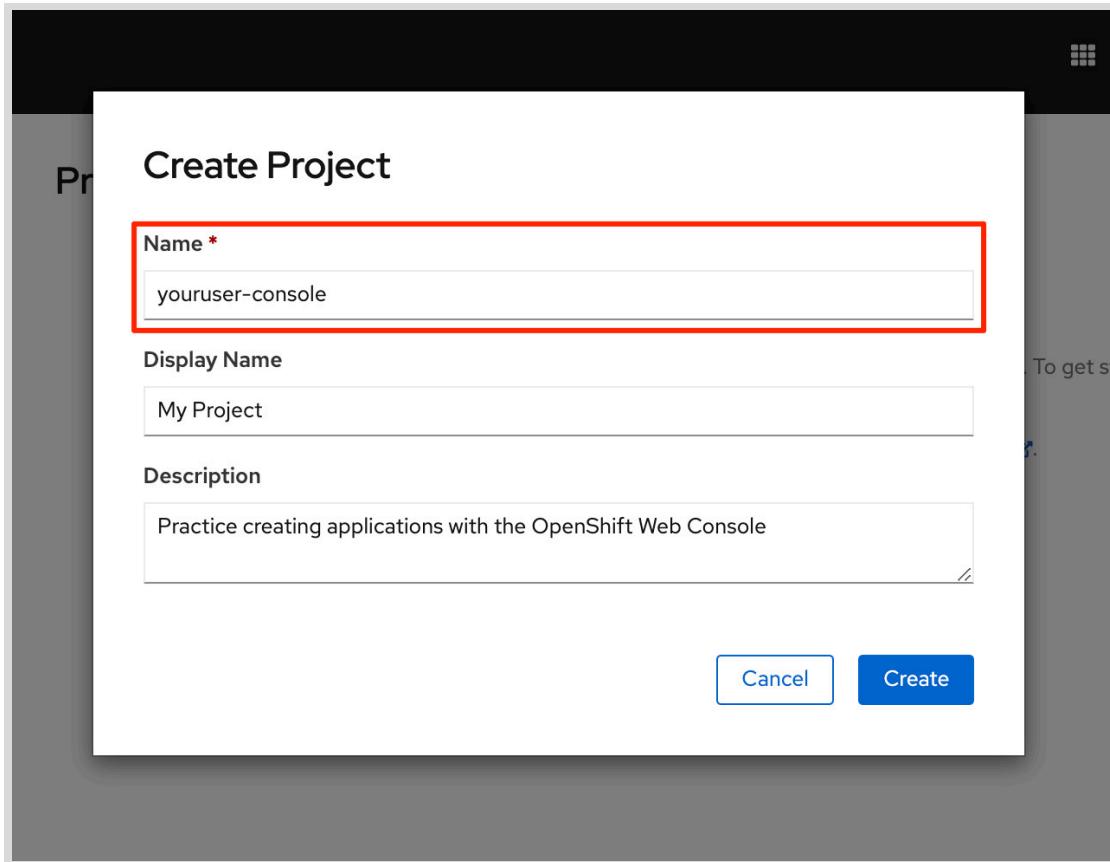


Figure 6.14: Création d'un projet

- 2.5. Après avoir rempli les champs obligatoires, cliquez sur **Create** dans la boîte de dialogue **Create Project** pour accéder à la page **Project Status** du projet **youruser-console**:

Details		Status		Activity	
Name	youruser-console	Active		View events	
Requester	youruser			Ongoing	
Labels	No labels			There are no ongoing activities.	
		Utilization		Recent Events	
		Resource	Usage	Pause	
		CPU		There are no recent events.	
		Memory			
		Filesystem			
		Inventory			
		0 Deployments			
		0 Deployment Configs			
		0 Stateful Sets			

Figure 6.15: Page Project Status

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

► 3. Créez l'application php-helloworld avec un modèle PHP.

- 3.1. Sélectionnez la perspective **Developer** dans la liste située en haut du menu de gauche.

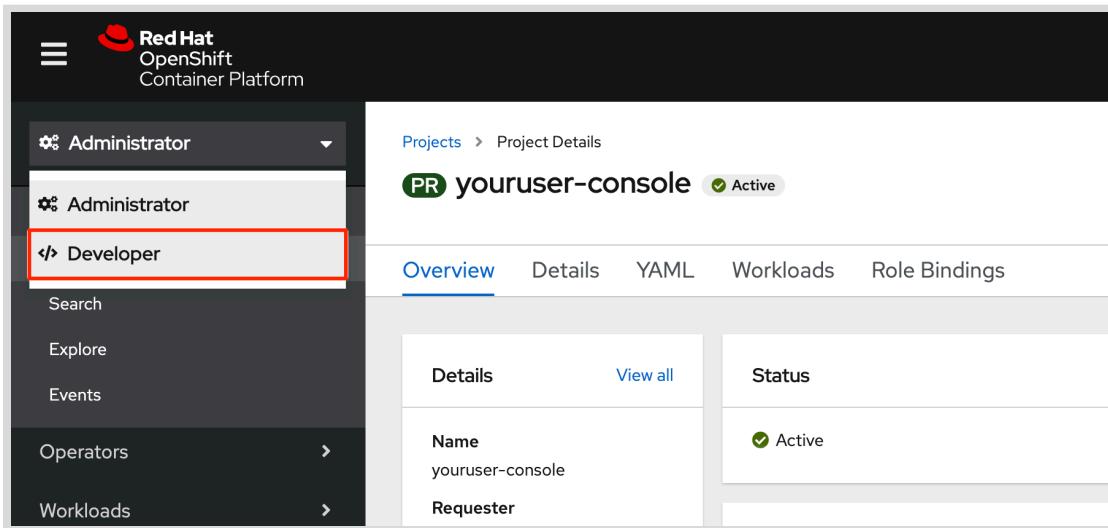


Figure 6.16: Liste déroulante de la perspective Developer

- 3.2. Cliquez sur **+Add** dans le menu gauche. Sur la page ouverte, cliquez sur **All services**, qui se trouve dans la section **Developer Catalog**, pour afficher une liste de modèles technologiques.

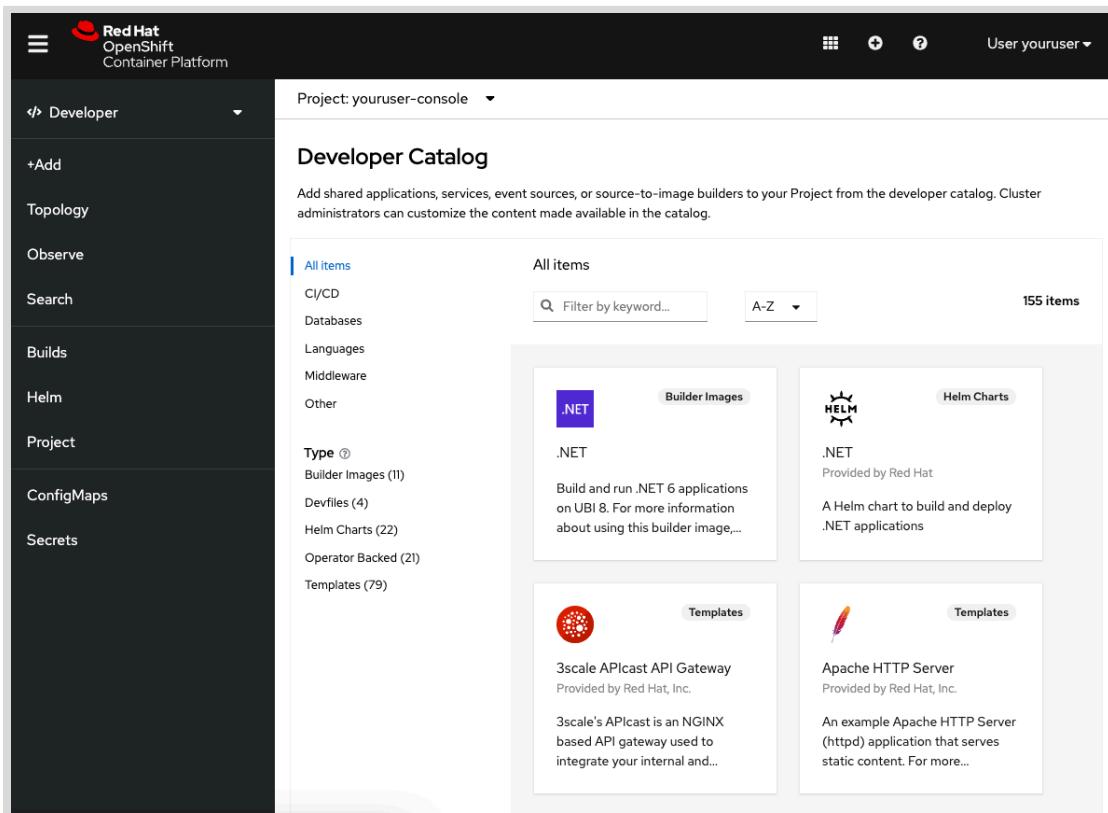


Figure 6.17: Page Developer Catalog

- 3.3. Saisissez **php** dans le champ **Filter by keyword**.

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, a sidebar menu includes options like Developer, Topology, Observe, Search, Builds, Helm, Project, ConfigMaps, and Secrets. The main area is titled "Developer Catalog" and displays a search bar with the term "php" entered. Below the search bar, there are two sections: "Templates" and "Builder Images". Under "Templates", there are two items: "CakePHP + MySQL" (Ephemeral) and "PHP". Under "Builder Images", there is one item: "PHP". Each item has a brief description and a link for more information. The top right corner of the main area shows "3 items".

Figure 6.18: Recherche de modèles liés à PHP

- 3.4. Après le filtrage, cliquez sur l'image de compilateur PHP pour afficher la boîte de dialogue PHP. Cliquez sur **Create Application** pour afficher la page **Create Source-to-Image Application**.

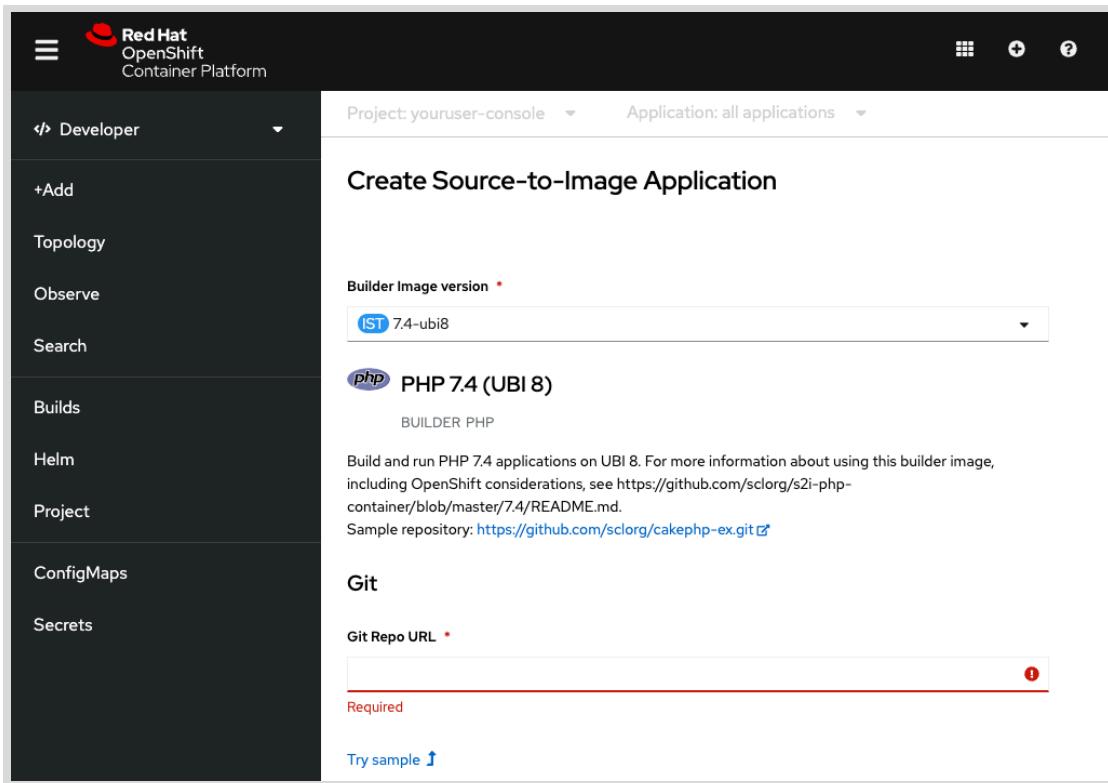


Figure 6.19: Configuration de Source-to-Image pour une application PHP

- 3.5. Définissez Builder Image Version sur la version 7.4 de PHP (UBI 8).
Spécifiez l'emplacement du référentiel Git du code source : <https://github.com/yourgituser/D0180-apps>
Utilisez Advanced Git Options pour définir le répertoire de contexte sur php-helloworld et la console de branche pour cet exercice.

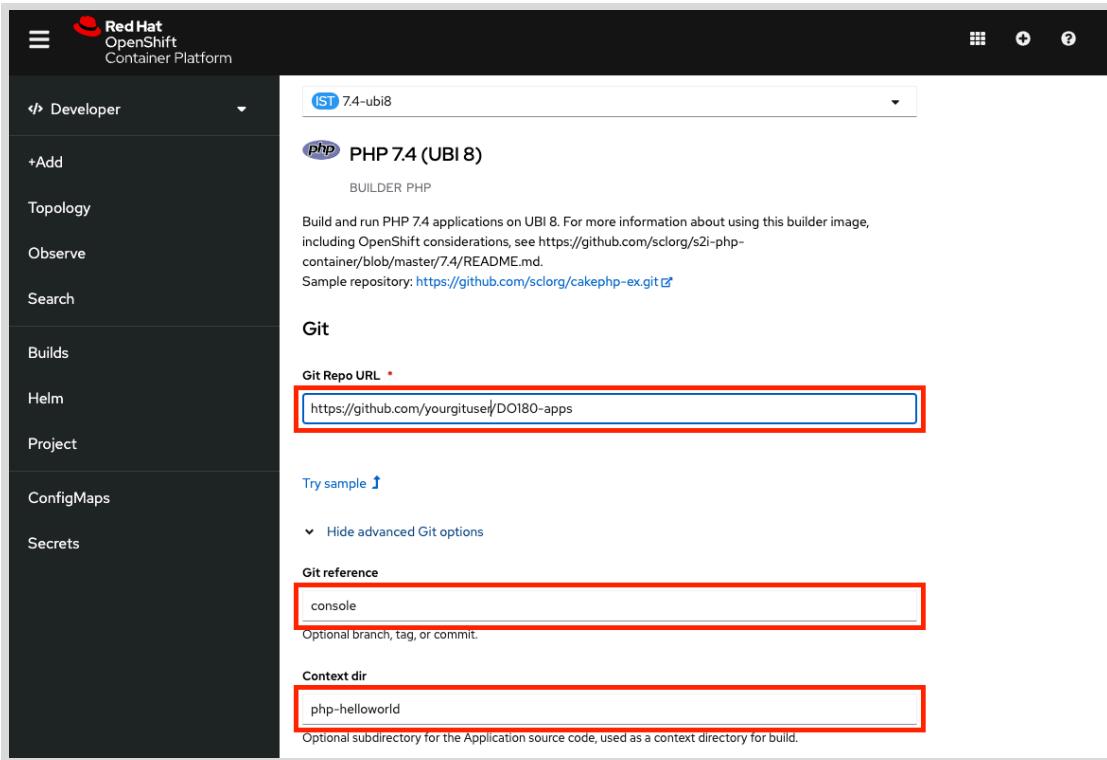


Figure 6.20: Définition des options Git avancées pour l'application

Saisissez php-HelloWorld pour le nom d'application et le nom utilisé pour les ressources associées. Sélectionnez **Deployment** comme type de ressource.

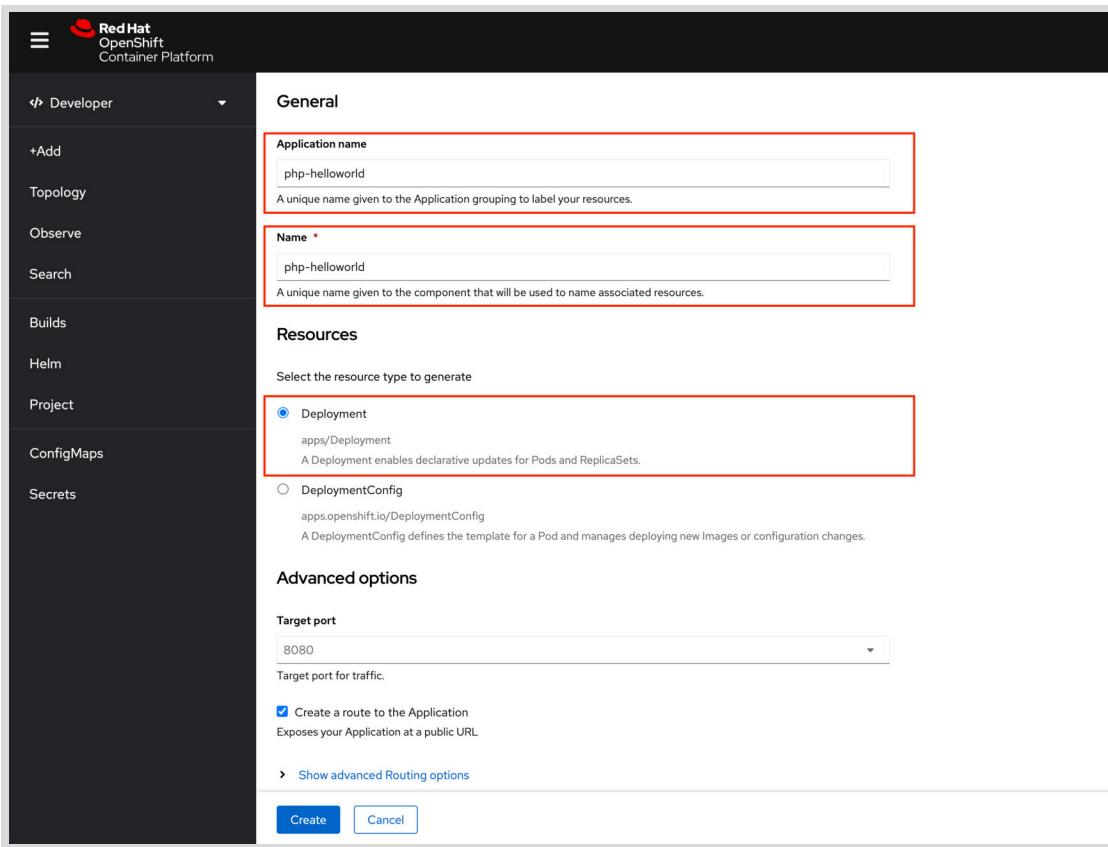


Figure 6.21: Définition des options d'application

Faites défiler vers le bas de la page et sélectionnez **Create a route to the application**. Cliquez sur **Create** pour créer les ressources OpenShift et Kubernetes requises pour l'application.

Vous êtes redirigé vers la page **Topology** :

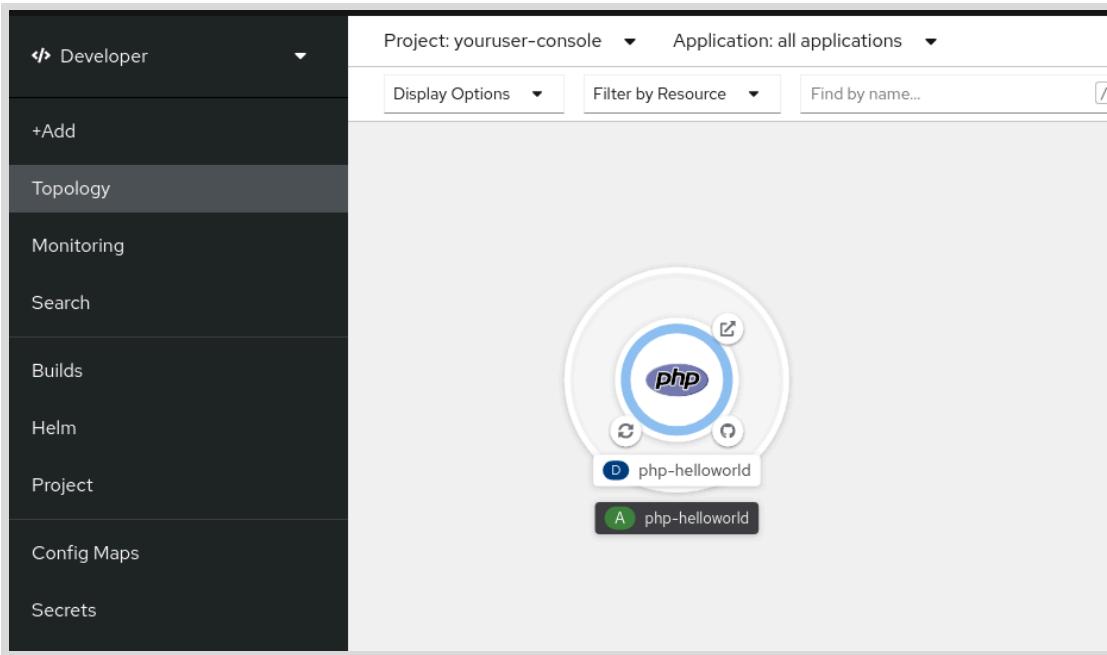


Figure 6.22: Page Topology

Cette page indique que l'application `php-helloworld` est créée. L'annotation D à gauche du lien `php-helloworld` est l'acronyme de Deployment. Ce lien redirige vers une page contenant des informations sur le déploiement de l'application.

3.6. Revenez à la perspective **Administrator** pour le reste de l'exercice :

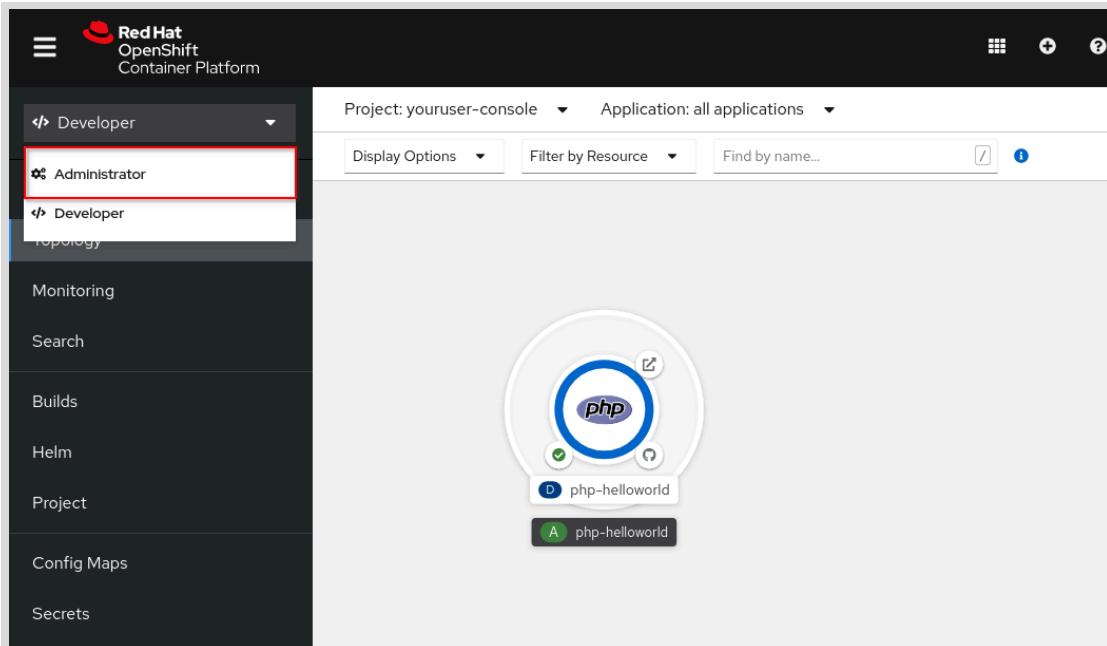


Figure 6.23: Liste déroulante de la perspective Administrator

- 4. Utilisez la barre de navigation située à gauche de la console Web OpenShift pour rechercher des informations sur les ressources OpenShift et Kubernetes de l'application :

- Déploiements

- BuildConfig
 - Journaux de compilation
 - Service
 - Route
- 4.1. Examinez le déploiement. Dans la barre de navigation, cliquez sur **Workloads > Deployments** pour afficher une liste de déploiements pour le projet *youruser-console*. Cliquez sur le lien **php-helloworld** pour afficher les informations de déploiement.

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, there is a sidebar with the following navigation items:

- Events
- Operators
- Workloads
 - Pods**
 - Deployments** (selected)
 - DeploymentConfigs
 - StatefulSets
 - Secrets
 - ConfigMaps
- CronJobs
- Jobs
- DaemonSets
- ReplicaSets
- ReplicationControllers
- HorizontalPodAutoscalers

Below the Workloads section, there are sections for Networking, Storage, and Builds.

The main content area displays the deployment details for the 'php-helloworld' deployment. It includes:

- Deployment details**: A circular icon indicating 1 Pod.
- Name**: php-helloworld
- Namespace**: jmzzjv-console
- Labels** (Edit): app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php-helloworld, app.kubernetes.io/part-of=php-helloworld, app.openshift.io/runtime=php, app.openshift.io/runtime-version=7.4-ubi8
- Update strategy**: RollingUpdate
- Max unavailable**: 25% of 1 pod
- Max surge**: 25% greater than 1 pod
- Progress deadline seconds**: 600 seconds
- Min ready seconds**: Not configured
- Pod selector**: Q app=php-helloworld

Figure 6.24: Page de détails du déploiement d'applications

Explorez les informations disponibles à partir de l'onglet **Details**. Il se peut que la compilation soit toujours en cours lorsque vous atteignez cette page ; il est donc possible que la valeur **1 pod** ne soit pas encore définie pour Deployment.

Si vous cliquez sur les icônes Flèche vers le haut et vers le bas en regard du graphique en anneau qui indique le nombre de pods, vous pouvez mettre à l'échelle l'application horizontalement.

- 4.2. Examinez la configuration de compilation. Dans la barre de navigation, cliquez sur **Builds > Build Configs** pour afficher une liste de configurations de build pour le projet *youruser-console*. Cliquez sur le lien **php-helloworld** pour afficher la configuration de compilation pour l'application.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar has sections for Jobs, DaemonSets, ReplicaSets, ReplicationControllers, and HorizontalPodAutoscalers. Below that are Networking (Services, Routes, Ingresses, NetworkPolicies), Storage, Builds (selected), Compute, and User Management. The main content area shows a 'BuildConfig details' page for 'BC php-helloworld'. The top navigation bar shows 'Project: youruser-console' and 'Actions'. The 'Details' tab is selected. The page displays the following configuration details:

Field	Value
Name	php-helloworld
Type	Source
Namespace	youruser-console
Git repository	https://github.com/yourgithubuser/DO180-apps
Labels	app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php-helloworld, app.kubernetes.io/part-of=php-helloworld, app.openshift.io/runtime=php, app.openshift.io/runtime-version=7.4-ubi8
Git ref	console
Context dir	php-helloworld
Build from	ISTI php:7.4-ubi8
Output to	ISTI php-helloworld:latest
Run policy	Serial
Triggers	Generic, GitHub, ImageChange, ConfigChange

Figure 6.25: Page de détails de la configuration de la compilation d'applications

Explorez les informations disponibles à partir de l'onglet **Details**. L'onglet **YAML** vous permet d'afficher et de modifier la configuration de compilation sous forme d'un fichier YAML. L'onglet **Builds** fournit une liste historique des compilations, ainsi qu'un lien vers plus d'informations pour chaque compilation. L'onglet **Environment** vous permet d'afficher et de modifier les variables d'environnement pour l'environnement de compilation de l'application. L'onglet **Events** affiche une liste d'événements et de métadonnées liés à la compilation.

- 4.3. Examinez les journaux pour la compilation Source-to-Image de l'application. In the **Builds** menu, click **Builds** to display a list of recent builds for the *youruser-console* project.

Cliquez sur le lien **php-helloworld-1** pour accéder aux informations pour la première compilation de l'application **php-helloworld**:

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, a sidebar lists various resources: Deployments, DeploymentConfigs, StatefulSets, Secrets, ConfigMaps, CronJobs, Jobs, DaemonSets, ReplicaSets, ReplicationControllers, and HorizontalPodAutoscalers. Under the 'Builds' section, 'Builds' is selected. The main content area is titled 'Build details' for 'php-helloworld-1'. The 'Details' tab is active, showing the following details:

- Name:** php-helloworld-1
- Status:** Complete
- Namespace:** youruser-console
- Type:** Source
- Git repository:** https://github.com/yourgithubuser/DO180-apps
- Git ref:** console
- Git commit:** Initial commit, including all apps previously in course
f7cd896 by Jordi Sola
- Context dir:** php-helloworld
- Build from:** image-registry.openshift-image-registry.svc:5000/openshift/php@sha256:696a1dc9240b68114cf5998412f116056ada4ffda2e9c9a6281e6eac7fcacf66

Other tabs include Metrics, YAML, Environment, Logs, and Events.

Figure 6.26: Page de détails de la compilation d'applications

Explorez les informations disponibles à partir de l'onglet Details. Ensuite, cliquez sur l'onglet Logs. Une zone de texte déroulante contient les résultats du processus de compilation :

The screenshot shows the Red Hat OpenShift Container Platform interface, similar to Figure 6.26. The sidebar and navigation are identical. The main content area is titled 'Build details' for 'php-helloworld-1'. The 'Logs' tab is active, showing a log stream with 63 lines of text. The log output is as follows:

```

Log stream ended. Search Debug container Wrap lines Raw Download Expand
63 lines
43 Copying blob sha256:f1fd656a1f6e7b6afb2b3523a01c717b13d94bd9dede6c1f0061ed9e70b22d65
44 Copying config sha256:98a358743a0541f78e13a5a14a9jaadd4fd4acd3505ab5d057fc274581887212
45 Writing manifest to image destination
46 Storing signatures
47 --> 98a358743a0
48 Successfully tagged temp.builder.openshift.io/jmzzjv-console/php-helloworld-1:3454a6ca
49 98a358743a0541f78e13a5a14a9jaadd4fd4acd3505ab5d057fc274581887212
50 Pushing image image-registry.openshift-image-registry.svc:5000/jmzzjv-console/php-helloworld:latest ...
51 Getting image source signatures
52 Copying blob sha256:5dcabd60ea6b60326f98e2b49d6ebcb771df4b70c6297ddf2d7dede6692df6e
53 Copying blob sha256:79a56ba04a301eb949644bca29f18b1879b6f305091ef1eb8068a0f5828db863
54 Copying blob sha256:867113e1c57d3106cae2383f9bbfe1c45a26eac03e82786a494e15956c3
55 Copying blob sha256:aad543859364662dd2b64ad5752fd9449d47410b9eafa02784630a9c578b79c6
56 Copying blob sha256:12bb1afe30f6fe8a264840ad3f250bb2b335b9ebca922db57b48836ceec08c
57 Copying blob sha256:aa0543859364662dd2b64ad5752fd9449d47410b9eafa02784630a9c578b79c6
58 Copying config sha256:98a358743a0541f78e13a5a14a9jaadd4fd4acd3505ab5d057fc274581887212
59 Writing manifest to image destination
60 Storing signatures
61 --> Push successful
62 Successfully pushed image-registry.openshift-image-registry.svc:5000/jmzzjv-console/php-helloworld@sha256:62
63

```

Figure 6.27: Journaux pour une compilation d'application

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

Lorsque Podman crée une image de conteneur, une sortie similaire est observée par rapport à la sortie affichée dans le navigateur.

- 4.4. Recherchez des informations pour le service de l'application `php-helloworld`. Dans la barre de navigation, cliquez sur **Networking > Services** pour afficher une liste de services pour le projet `youruser-console`. Cliquez sur le lien `php-helloworld` pour afficher les informations associées au service de l'application :

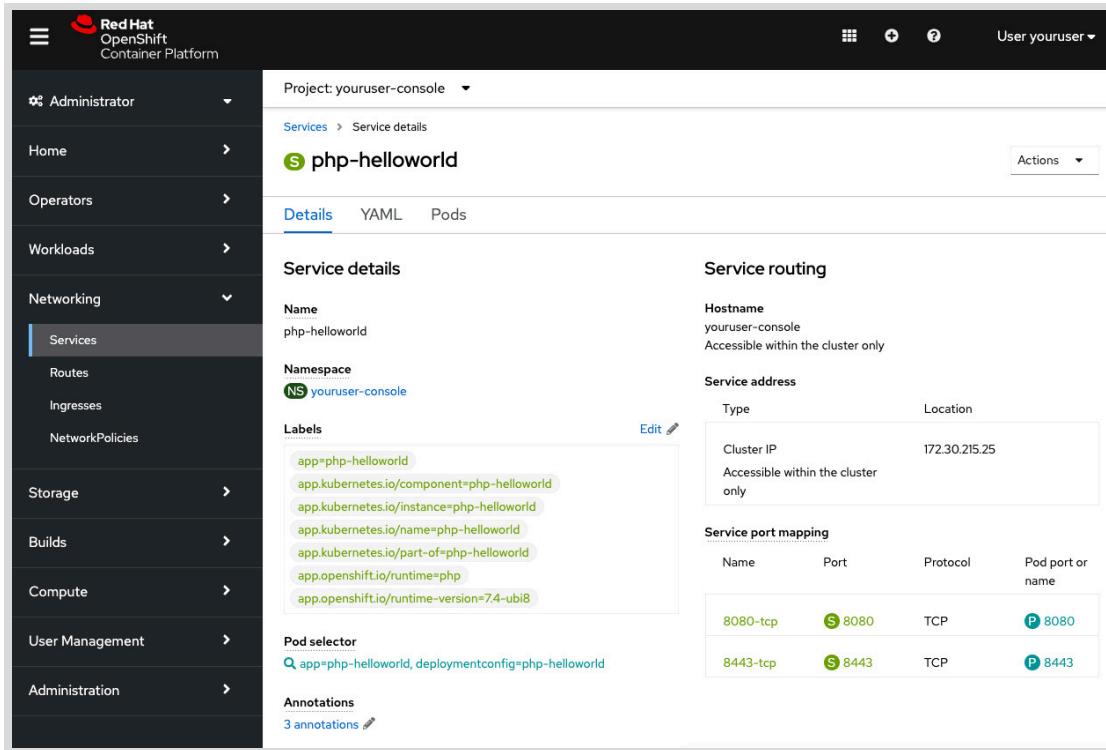


Figure 6.28: Page de détails du service

Explorez les informations disponibles à partir de l'onglet **Details**. L'onglet **YAML** vous permet d'afficher et de modifier la configuration du service, sous forme d'un fichier YAML. L'onglet **Pods** affiche la liste actuelle des pods qui fournissent le service d'application.

- 4.5. Localisez les informations de routage externe pour l'application. Dans la barre de navigation, cliquez sur **Networking > Routes** pour afficher une liste de routes configurées pour le projet `youruser-console`. Cliquez sur le lien `php-helloworld` pour afficher les informations associées à la route de l'application :

The screenshot shows the Red Hat OpenShift web interface. The left sidebar has a dark theme with categories like Home, Operators, Workloads, Networking, Storage, Builds, Compute, User Management, and Administration. Under Networking, 'Routes' is selected. The main content area is titled 'Route details' for 'Route: php-helloworld'. It shows the route's name, namespace ('youruser-console'), labels (including app=php-helloworld), annotations (1 annotation), service ('php-helloworld'), location ('https://php-helloworld-youruser-console.apps.cluster.lab.example.com'), status ('Accepted'), host ('php-helloworld-youruser-console.apps.cluster.lab.example.com'), path ('-'), and router canonical hostname ('router-default.apps.cluster.lab.example.com'). An 'Actions' button is at the top right.

Figure 6.29: Page de détails de la route

Explorez les informations disponibles à partir de l'onglet Details. Le champ Location fournit un lien vers la route externe pour l'application ; `https://php-helloworld-$\{RHT_OCP4_DEV_USER\}-console.\$\{RHT_OCP4_WILDCARD_DOMAIN\}`. Cliquez sur le lien pour accéder à l'application dans un nouvel onglet :

The screenshot shows a Mozilla Firefox browser window. The address bar contains the URL 'php-helloworld-console.apps.cluster.lab.example.com'. The main content area displays the text 'Hello, World! php version is 7.4.19'.

Figure 6.30: Résultats initiaux de l'application PHP

- 5. Modifiez le code d'application, validez la modification, envoyez le code au référentiel Git distant, puis déclenchez une nouvelle build d'application.

- 5.1. Saisissez le répertoire du code source :

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

- 5.2. Ajoutez la deuxième instruction de ligne sur la page `index.php` pour qu'elle indique « A change is in the air! », puis enregistrez le fichier. Ajoutez la modification à l'index Git, validez-la, puis envoyez les modifications au référentiel Git distant.

```
[student@workstation php-helloworld]$ vim index.php
[student@workstation php-helloworld]$ cat index.php
<?php
print "Hello, World! php version is " . PHP_VERSION . "\n";
print "A change is in the air!\n";
?>
[student@workstation php-helloworld]$ git add index.php
[student@workstation php-helloworld]$ git commit -m 'updated app'
[console d198fb5] updated app
...output omitted...
1 file changed, 1 insertion(+), 1 deletion(-)
[student@workstation php-helloworld]$ git push origin console
Counting objects: 7, done.
Delta compression using up to 2 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 409 bytes | 0 bytes/s, done.
Total 4 (delta 1), reused 0 (delta 0)
...output omitted...
```

5.3. Déclenchez une build d'application manuellement à partir de la console Web.

Dans la barre de navigation, cliquez sur **Builds > Build Configs**, puis cliquez sur le lien **php-helloworld** pour accéder à la page **Build Config Details**. Dans le menu Actions en haut à droite de l'écran, cliquez sur **Start Build** :

The screenshot shows the Red Hat OpenShift web interface. The left sidebar has a dark theme with categories like Administrator, Home, Operators, Workloads, Networking, Storage, Builds, Compute, and User Management. Under Builds, 'BuildConfigs' is selected. The main panel shows a project named 'youruser-console'. In the center, there's a 'BuildConfigs > BuildConfig details' section for 'BC php-helloworld'. The 'Details' tab is selected. On the right, there's a 'Actions' dropdown menu with several options: Start build (which is highlighted with a red box), Edit labels, Edit annotations, Edit BuildConfig, and Delete BuildConfig. Below the actions, there are sections for 'BuildConfig details' (Name: php-helloworld, Type: Source, Namespace: youruser-console, Git repository: https://github.com/yourgithubuser/DO180-apps), 'Labels' (with several annotations listed), 'Git ref' (set to 'console'), 'Context dir' (set to 'php-helloworld'), 'Build from' (set to 'ISI php:7.4-ubi8'), 'Output to' (set to 'ISI php-helloworld:latest'), 'Annotations' (3 annotations listed), 'Run policy' (set to 'Serial'), and 'Triggers' (set to 'Generic, GitHub, ImageChange, ConfigChange'). At the bottom, there's a 'Created at' timestamp: Feb 25, 2022, 1:14 PM.

Figure 6.31: Lancement d'une compilation d'application

chapitre 6 | Déploiement d'applications en conteneur dans OpenShift

Vous êtes redirigé vers une page Build Details pour la nouvelle compilation. Cliquez sur l'onglet **Logs** pour surveiller l'avancement de la compilation. La dernière ligne d'une compilation réussie contient `Push successful`.

Une fois la compilation terminée, le déploiement commence. Accédez à la section **Workloads > Pods** et attendez que le nouveau pod soit déployé et en cours d'exécution.

- 5.4. Rechargez l'URL `http://php-helloworld-${RHT_OCP4_DEV_USER}-console.${RHT_OCP4_WILDCARD_DOMAIN}` dans le navigateur. La réponse de l'application correspond au code source mis à jour :



Figure 6.32: Sortie de l'application Web mise à jour

6. Supprimez le projet. Dans la barre de navigation, cliquez sur **Home > Projects**. Cliquez sur l'icône située à droite de la ligne contenant une entrée pour le projet `youruser-console`. Cliquez sur **Delete Project** dans le menu qui apparaît.

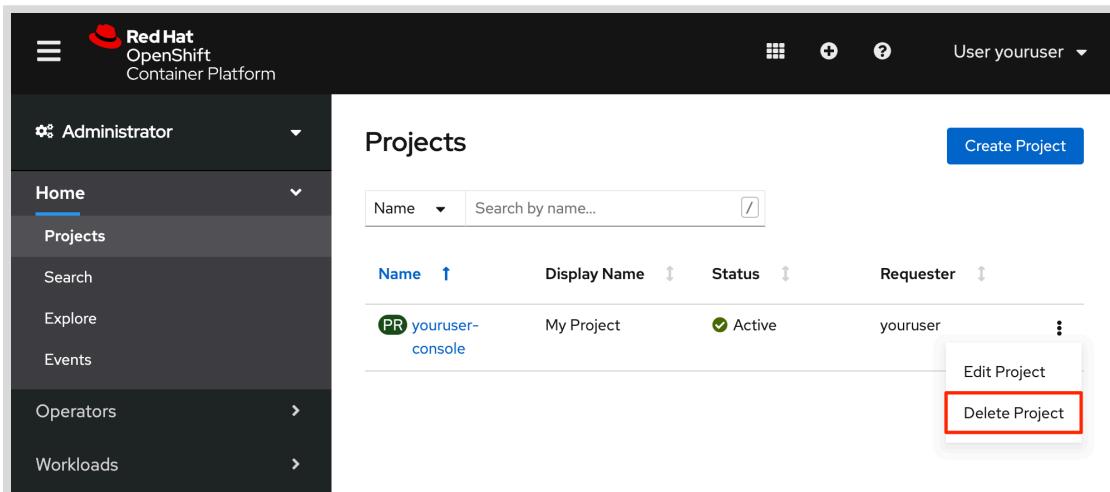


Figure 6.33: Suppression d'un projet

Saisissez `youruser-console` dans la boîte de dialogue de confirmation, puis cliquez sur **Delete**.

Fin

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

```
[student@workstation php-helloworld]$ lab openshift-webconsole 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 `${RHT_OCP4_DEV_USER} - ocp`.
2. Créez une application de conversion de température nommée `temps` écrite en `temps` à l'aide de la balise de flux d'images `php:7.3`. Le code source se trouve dans le référentiel Git sous <https://github.com/RedHatTraining/D0180-apps/> dans le répertoire `temps`. Vous pouvez utiliser l'interface de ligne de commande ou la console Web OpenShift pour créer l'application.
3. Vérifiez que vous pouvez accéder à l'application dans un navigateur Web à l'adresse [http://temps- \\${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.

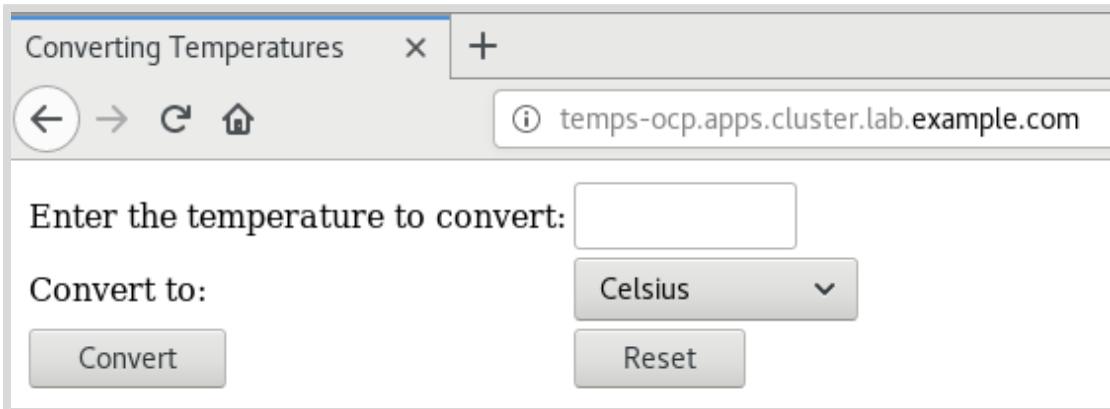
```
[student@workstation ~]$ oc expose svc/temps
route.route.openshift.io/temps exposed
```

3. Vérifiez que vous pouvez accéder à l'application dans un navigateur Web à l'adresse `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.
- Déployer une application multiconteneur sur OpenShift.
- Déployer une application sur OpenShift à l'aide d'un modèle.

Sections

- Considérations liées aux applications multiconteneur (avec exercice guidé)
- Déploiement d'une application multiconteneur sur OpenShift (et exercice guidé)
- Déploiement d'une application multiconteneur sur OpenShift à l'aide d'un modèle (et exercice guidé)

Atelier

- Déploiement d'applications multiconteneurs

Considérations liées aux applications multiconteneurs

Résultats

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

- Décrire les considérations relatives à la conteneurisation des applications avec plusieurs images de conteneur.
- Exploiter les concepts de mise en réseau dans des conteneurs.
- Créer une application multiconteneur avec Podman.
- Décrivez l'architecture de l'application To Do List.

Exploitation d'applications multiconteneurs

Les exemples présentés jusqu'à présent tout au long de ce cours ont bien fonctionné avec un seul conteneur. Toutefois, une application plus complexe peut tirer parti du déploiement de différents composants dans différents conteneurs. Considérons une application composée d'une application Web frontale, d'un backend REST et d'un serveur de base de données. Ces composants peuvent avoir des dépendances, des exigences et des cycles de vie différents.

Bien qu'il soit possible d'orchestrer des conteneurs d'applications multiconteneurs manuellement, Kubernetes et OpenShift fournissent des outils permettant de faciliter l'orchestration. Tenter de gérer manuellement des dizaines ou des centaines de conteneurs devient rapidement compliqué. Dans cette section, nous allons revenir à l'utilisation de Podman pour créer une application multiconteneur simple visant à illustrer les étapes manuelles sous-jacentes à l'orchestration de conteneurs. Dans les sections suivantes, vous allez utiliser Kubernetes et OpenShift pour orchestrer ces mêmes conteneurs d'applications.

Découverte de services dans une application multiconteneur

Conteneurs avec racine

Podman utilise Container Network Interface (CNI) pour créer un réseau de type Software-Defined (SDN) entre tous les conteneurs de l'hôte. Sauf indication contraire, CNI attribue une nouvelle adresse IP à un conteneur lors de son démarrage.

Chaque conteneur expose l'ensemble des ports aux autres conteneurs du même réseau SDN. Dès lors, les services sont facilement accessibles au sein du même réseau. Les conteneurs n'exposent les ports aux réseaux externes que selon une configuration explicite.

La nature dynamique des adresses IP de conteneur empêche les applications de s'appuyer sur des adresses IP fixes ou des noms d'hôtes DNS fixes pour communiquer avec les services de Middleware et d'autres services d'application. Les conteneurs présentant des adresses IP dynamiques peuvent poser problème lorsque vous utilisez des applications multiconteneurs parce que chaque conteneur doit pouvoir communiquer avec les autres afin d'utiliser les services dont il dépend.

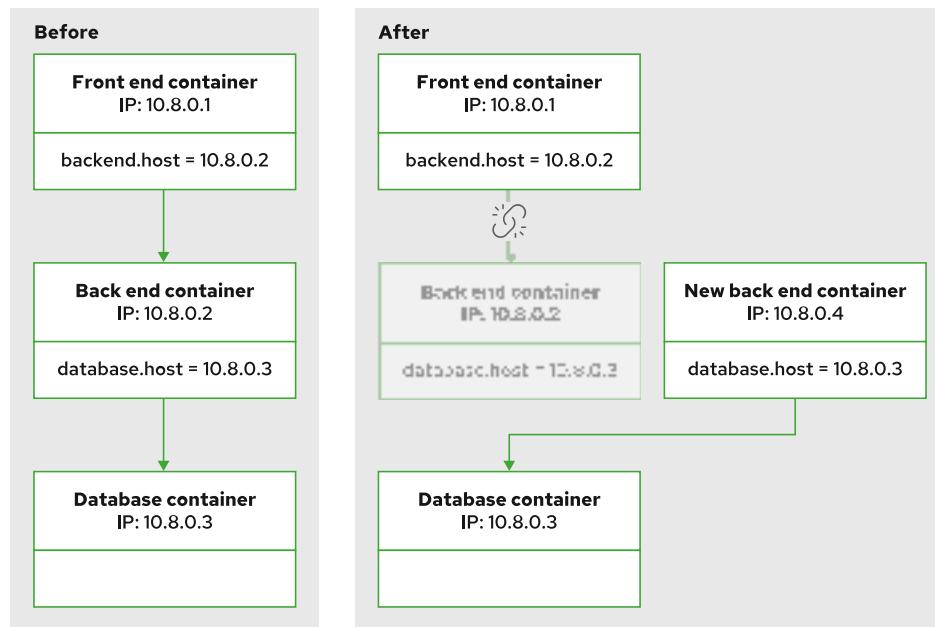


Figure 7.1: Un redémarrage rompt les liens d'application à trois niveaux

Par exemple, prenons une application composée d'un conteneur frontal, d'un conteneur backend et d'une base de données. Le conteneur frontal doit récupérer l'adresse IP du conteneur backend. De la même façon, le conteneur backend doit récupérer l'adresse IP du conteneur de base de données. De plus, l'adresse IP peut être modifiée en cas de redémarrage d'un conteneur. C'est pourquoi un processus est nécessaire pour garantir que toute modification de l'adresse IP déclenche une mise à jour des conteneurs existants.

Kubernetes et OpenShift permettent de résoudre les problèmes liés à la découverte des services et à la nature dynamique de la mise en réseau des conteneurs. Certaines de ces solutions sont abordées plus tard dans le chapitre.

Conteneurs sans racine

Les conteneurs sans racine ne prennent pas en charge le réseau défini par logiciel (SDN, Software-Defined Network). Par conséquent, l'adresse IP du conteneur n'est pas disponible pour communiquer avec d'autres conteneurs sur l'hôte. Nous pouvons établir la mise en réseau entre les conteneurs sans racine en utilisant le réacheminement de ports. Le réacheminement de ports permet un accès externe à un service de conteneur à partir de l'hôte. Le réacheminement de ports a été abordé dans *Chapitre 3, Gestion des conteneurs*.

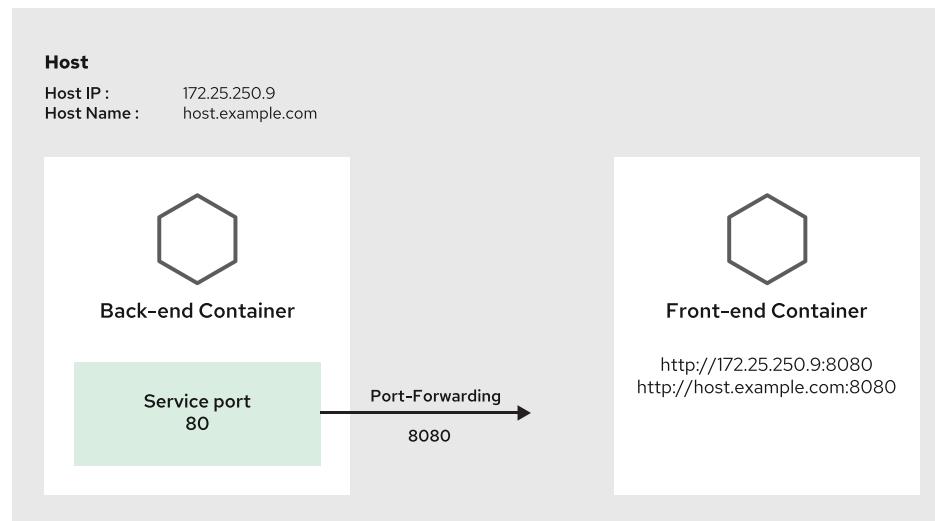


Figure 7.2: Applications multiconteneurs pour les conteneurs sans racine

Par exemple, prenons une application présentant à la fois un conteneur frontal et un conteneur backend. Le conteneur backend dispose d'un service exécuté sur le port spécifique 80. Ce service/port n'est pas accessible depuis l'extérieur du conteneur. Le conteneur frontal doit accéder au service/port à partir du conteneur backend. La première étape consiste à utiliser le *réacheminement de ports* pour le service requis. Dans cet exemple, nous avons transféré le port de service 80 à 8080. À présent, le conteneur frontal peut accéder au service/port du conteneur backend en utilisant l'adresse IP de l'hôte et le port transféré.

Description de l'application To Do List

De nombreux ateliers dans ce cours recourent à une application To Do List. Cette application comprend trois niveaux, comme illustré par la figure suivante :

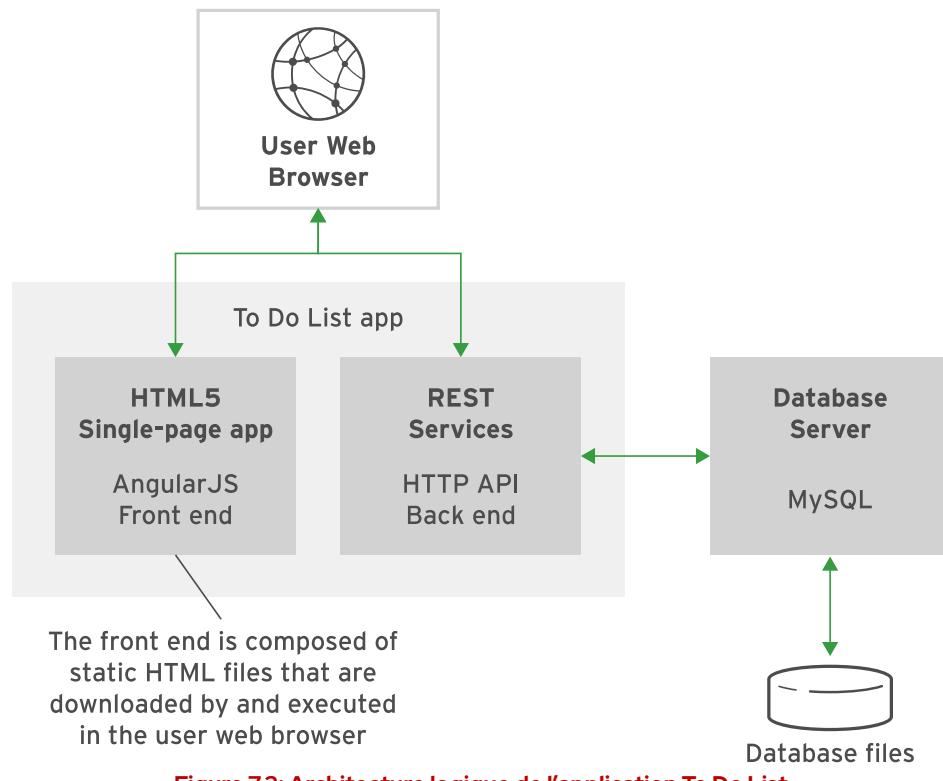


Figure 7.3: Architecture logique de l'application To Do List

- Le niveau de présentation est conçu comme une page unique HTML5 frontale utilisant AngularJS.
- Le niveau métier d'un backend API HTTP avec Node.js.
- Le niveau de persistance est basé sur un serveur de base de données MySQL.

L'illustration suivante représente une capture d'écran de l'interface Web de l'application :

To Do List Application

To Do List

Id	Description	Done	
1	Pick up new...	false	X
2	Buy groceries	true	X

[First](#) [Previous](#) [1](#) [Next](#) [Last](#)

Add Task

Description:

Add Description.

Completed:

[Clear](#) [Save](#)

Figure 7.4: L'application To Do List

À gauche se trouve un tableau avec des tâches à effectuer, et à droite, un formulaire permet d'ajouter une nouvelle tâche.

Le serveur de registre privé de la salle de classe, `services.lab.example.com`, fournit deux versions pour l'application :

nodejs

Représente la façon dont un développeur classique créerait l'application en tant qu'unité simple, sans aucune division en niveaux ou services.

nodejs_api

Affiche les modifications nécessaires pour scinder l'application en niveaux de présentation et métier. Chaque niveau correspond à une image de conteneur isolée

Les sources de ces deux versions d'application sont disponibles à partir du dossier `todoapp/nodejs` dans le référentiel Git à l'adresse suivante : <https://github.com/RedHatTraining/DO180-apps.git>.

► Exercice guidé

Déploiement de l'application Web et de conteneurs MySQL sur Linux

Dans cet atelier, vous allez créer un script qui exécute et met en réseau un conteneur d'application Node.js et le conteneur MySQL.

Résultats

Vous pourrez mettre des conteneurs en réseau afin de créer une application multicouche.

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 configurer l'environnement pour cet exercice, exécutez la commande suivante :

```
[student@workstation ~]$ lab multicontainer-design start
```

Instructions

- 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 D, Création d'un compte Red Hat*.

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

- 2. Examinez le Containerfile.

À l'aide de votre éditeur préféré, ouvrez et examinez le Containerfile terminé situé à l'adresse `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/Containerfile`

- 3. Exécutez la commande `ip addr` vers l'adresse IP de l'hôte grep.

```
[student@workstation ~]$ ip -br addr list | grep eth0
eth0          UP              172.25.250.9/24 fe80::d1cf:b1dc:ddc8:b79b/64
```

- 4. Explorez les variables d'environnement.

Examinez les variables d'environnement qui permettent au conteneur Node.js de l'API REST de communiquer avec le conteneur MySQL.

- 4.1. Affichez le fichier `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/nodejs-source/models/db.js`, contenant la configuration de base de données fournie ci-dessous :

```
module.exports.params = {  
  dbname: process.env.MYSQL_DATABASE,  
  username: process.env.MYSQL_USER,  
  password: process.env.MYSQL_PASSWORD,  
  params: {  
    host: '172.25.250.9',  
    port: '30306',  
    dialect: 'mysql'  
  }  
};
```

- 4.2. Notez les variables d'environnement utilisées par l'API REST. Ces variables sont exposées au conteneur à l'aide des options `-e` avec la commande `podman run` de cet exercice guidé. Ces variables d'environnement sont décrites ci-dessous.

MYSQL_DATABASE

Nom de la base de données MySQL dans le conteneur `mysql`.

MYSQL_USER

Nom de l'utilisateur de base de données employé par le conteneur `todoapi` pour exécuter des commandes MySQL.

MYSQL_PASSWORD

Mot de passe de l'utilisateur de base de données employé par le conteneur `todoapi` pour s'authentifier auprès du conteneur `mysql`.



Note

Les informations sur l'hôte et le port du conteneur MySQL sont intégrées dans l'application API REST. L'hôte, comme illustré ci-dessus dans le fichier `db.js` est l'adresse IP de host, pour laquelle nous avons utilisé grep à l'étape précédente.

- ▶ 5. Compilez l'image enfant de l'application `To Do List` à l'aide du fichier `Containerfile` fourni.

- 5.1. Compilez l'image enfant.

Examinez le script `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/build.sh` pour voir comment l'image est créée. Exécutez les commandes suivantes pour compiler l'image enfant.

```
[student@workstation nodejs]$ cd ~/D0180/labs/multicontainer-design/deploy/nodejs  
[student@workstation nodejs]$ ./build.sh  
Preparing build folder  
Preparing build folder  
STEP 1: FROM registry.redhat.io/rhel8/nodejs-12  
Getting image source signatures  
Copying blob 666be21779ed done  
...output omitted...  
Writing manifest to image destination  
Storing signatures
```

```
STEP 2: ARG NEXUS_BASE_URL
STEP 3: MAINTAINER username <username@example.com>
STEP 4: COPY run.sh build ${HOME}/
STEP 5: RUN npm install --registry=http://$NEXUS_BASE_URL/repository/nodejs/
...output omitted...
Writing manifest to image destination
Storing signatures
e4901...d37eb
```



Note

Le script `build.sh` réduit les restrictions d'accès en écriture au répertoire de compilation, permettant ainsi l'installation de dépendances par des utilisateurs non root.

Parfois, l'instabilité du réseau provoque une `ERR!` avec le script de compilation. Pour résoudre ce problème, exécutez à nouveau le script de compilation.

- 5.2. Attendez que la compilation soit terminée, puis exécutez la commande suivante pour vérifier que l'image a été compilée avec succès :

```
[student@workstation nodejs]$ podman images \
> --format "table {{.ID}} {{.Repository}} {{.Tag}}"
IMAGE ID      REPOSITORY          TAG
e4901b30413b  localhost/do180/todonodejs    latest
6b949cc5e908  registry.redhat.io/rhel8/nodejs-12  1
```

- ▶ 6. Modifiez le script existant pour créer des conteneurs avec les ports appropriés, comme défini à l'étape précédente. Dans ce script, l'ordre des commandes est indiqué de telle sorte qu'il démarre le conteneur `mysql`, puis le conteneur `todoapi` avant de le connecter au conteneur `mysql`. Un temps d'attente de 9 secondes est observé une fois que tous les conteneurs ont été appelés, de façon à ce que chacun d'eux ait le temps de démarrer.
- 6.1. Modifiez le fichier `run.sh` situé sur `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/networked` afin d'insérer la commande `podman run` à la ligne appropriée pour appeler le conteneur `mysql`. La commande `podman` exacte à insérer dans le fichier est présentée dans l'écran suivant.

```
podman run -d --name mysql -e MYSQL_DATABASE=items -e MYSQL_USER=user1 \
-e MYSQL_PASSWORD=mypassword -e MYSQL_ROOT_PASSWORD=r00tpass \
-v $PWD/work/data:/var/lib/mysql/data \
-p 3306:3306 \
registry.redhat.io/rhel8/mysql-80:1
```

Dans la commande précédente, `MYSQL_DATABASE`, `MYSQL_USER` et `MYSQL_PASSWORD` sont renseignés avec les informations d'identification permettant d'accéder à la base de données MySQL. Ces variables d'environnement sont obligatoires pour l'exécution du conteneur `mysql`. En outre, le dossier local `$PWD/work/data` est monté en tant que volume dans le système de fichiers du conteneur.

- 6.2. Dans le même fichier `run.sh`, insérez une autre commande `podman run` à la ligne appropriée afin d'exécuter le conteneur `todoapi`. La commande `docker` à insérer dans le fichier est présentée dans l'écran suivant.

```
podman run -d --name todoapi -e MYSQL_DATABASE=items -e MYSQL_USER=user1 \
-e MYSQL_PASSWORD=mypa55 \
-p 30080:30080 \
do180/todonodejs
```

**Note**

Après chaque commande `podman run` insérée dans le script `run.sh`, vérifiez qu'il y a également une commande `sleep 9`. Si vous devez répéter cette étape, supprimez le répertoire `work` et son contenu avant d'exécuter à nouveau le script `run.sh`.

- 6.3. Vérifiez que votre script `run.sh` correspond au script de solution situé sur `/home/student/D0180/solutions/multicontainer-design/deploy/nodejs/networked/run.sh`.
 - 6.4. Enregistrez le fichier et quittez l'éditeur.
- 7. Exédez les conteneurs.
- 7.1. Utilisez la commande suivante pour exécuter le script que vous avez mis à jour pour exécuter les conteneurs `mysql` et `todoapi`.

```
[student@workstation nodejs]$ cd \
> /home/student/D0180/labs/multicontainer-design/deploy/nodejs/networked
[student@workstation networked]$ ./run.sh
```

- 7.2. Vérifiez que les conteneurs ont bien démarré.
- ```
[student@workstation networked]$ podman ps \
> --format="table {{.ID}} {{.Names}} {{.Image}} {{.Status}}"
ID Names Image Status
c74b4709e3ae todoapi localhost/do180/todonodejs:latest Up 3 minutes ago
3bc19f74254c mysql registry.redhat.io/rhel8/mysql-80:1 Up 3 minutes ago
```
- 8. Renseignez la base de données `items` avec la table `Projects`.

```
[student@workstation networked]$ mysql -uuser1 -h 172.25.250.9 \
> -pmypa55 -P30306 items < \
> /home/student/D0180/labs/multicontainer-design/deploy/nodejs/networked/db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- 9. Examinez les variables d'environnement du conteneur de l'API.
- Exédez la commande suivante pour explorer les variables d'environnement exposées dans le conteneur de l'API.

```
[student@workstation networked]$ podman exec -it todoapi env
...output ommited...
HOME=/opt/app-root/src
MYSQL_DATABASE=items
MYSQL_USER=user1
MYSQL_PASSWORD=mypassword
APP_ROOT=/opt/app-root
```

► **10.** Testez l'application.

- 10.1. Exécutez une commande `curl` afin de tester l'API REST pour l'application To Do List.

```
[student@workstation networked]$ curl -w "\n" \
> http://127.0.0.1:30080/todo/api/items/1
{"id":1,"description":"Pick up newspaper","done":false}
```

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.

- 10.2. Ouvrez Firefox sur `workstation` et accédez à `http://127.0.0.1:30080/todo/` via votre navigateur. L'application To Do List doit normalement s'afficher.



**Note**

Prenez soin d'ajouter la barre oblique (/) à la fin.

- 10.3. Accédez au répertoire `/home/student/`.

```
[student@workstation networked]$ cd ~
[student@workstation ~]$
```

## Fin

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

```
[student@workstation ~]$ lab multicontainer-design finish
```

L'exercice guidé est maintenant terminé.

# 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.
- *Snakecase* : 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.

Ce sont les variables d'environnement générées par Kubernetes pour un service :

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".
```

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 `/home/student/D0180/labs/multicontainer-application/todo-`

app.yaml. Notez les ressources suivantes définies dans le todo-app.yaml 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 /home/student/D0180/labs/multicontainer-application
[student@workstation multicontainer-application]$ oc create -f todo-app.yaml
pod/mysql created
pod/todoapi created
service/todoapi created
service/mysql created
persistentvolumeclaim/dbclaim created
```

1.5. Examinez le déploiement.

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 /home/student/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}
```

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. Accédez au répertoire `/home/student/`.

```
[student@workstation multicontainer-application]$ cd ~
[student@workstation ~]$
```

- 4.4. Ouvrez Firefox sur `workstation` et saisissez l'adresse `http://todoapi-${RHT_OCP4_DEV_USER}-application.${RHT_OCP4_WILDCARD_DOMAIN}/todo/` : vous devez voir l'application `To Do List`.

**Note**

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 | X |
| 2  | Buy groceries  | true  | X |

First Previous **1** Next Last

#### Add Task

Description:

Add Description.

Completed:

**Clear** **Save**

Figure 7.5: 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.

# Déploiement d'une application multiconteneur sur OpenShift à l'aide d'un modèle

À la fin de cette section, les stagiaires seront en mesure de déployer une application multiconteneur sur OpenShift en utilisant un modèle.

## Examen du squelette d'un modèle

Le déploiement d'une application sur OpenShift Container Platform nécessite souvent la création de plusieurs ressources associées au sein d'un projet. Par exemple, une application Web peut nécessiter une ressource `BuildConfig`, `Deployment`, `Service` et `Route` pour s'exécuter dans un projet OpenShift. Souvent, les attributs de ces ressources ont la même valeur, comme l'attribut de nom d'une ressource.

Les modèles OpenShift fournissent un moyen de simplifier la création des ressources nécessaires à une application. Un modèle définit un ensemble de ressources associées à créer ensemble, ainsi qu'un ensemble de paramètres d'application. Les attributs des ressources de modèle sont généralement définis en termes de paramètres de modèle, tels que l'attribut de nom d'une ressource.

Par exemple, une application peut être composée d'une application Web frontale et d'un serveur de base de données. Chacune d'entre elles se compose d'une ressource de service et d'une ressource de déploiement. Elles partagent le même jeu d'informations d'identification (paramètres) pour que le système frontal se connecte au backend. Le modèle peut être traité en spécifiant ses paramètres ou en autorisant leur génération automatique (par exemple, pour un mot de passe de base de données unique) afin d'instancier la liste des ressources dans le modèle sous forme d'une application unifiée.

Le programme d'installation d'OCP crée plusieurs modèles par défaut dans l'espace de noms `openshift`. Exécutez la commande `oc get templates` avec l'option `-n openshift` pour lister ces modèles préinstallés :

```
[user@host ~]$ oc get templates -n openshift
NAME DESCRIPTION
cakephp-mysql-example An example CakePHP application ...
cakephp-mysql-persistent An example CakePHP application ...
dancer-mysql-example An example Dancer application with a MySQL ...
dancer-mysql-persistent An example Dancer application with a MySQL ...
django-psql-example An example Django application with a PostgreSQL ...
...output omitted...
rails-psql-persistent An example Rails application with a PostgreSQL ...
rails-postgresql-example An example Rails application with a PostgreSQL ...
redis-ephemeral Redis in-memory data structure store, ...
redis-persistent Redis in-memory data structure store, ...
```

La définition YAML d'un modèle peut être modifiée pour répondre aux besoins de vos applications. Vous allez le faire dans un prochain atelier.

Voici un exemple de définition de modèle YAML :

```
[user@host ~]$ oc get template mysql-persistent -n openshift -o yaml
apiVersion: template.openshift.io/v1
kind: Template
labels: ...value omitted...
message: ...message omitted ...
metadata:
 annotations:
 description: ...description omitted...
 iconClass: icon-mysql-database
 openshift.io/display-name: MySQL
 openshift.io/documentation-url: ...value omitted...
 openshift.io/long-description: ...value omitted...
 openshift.io/provider-display-name: Red Hat, Inc.
 openshift.io/support-url: https://access.redhat.com
 tags: database,mysql ①
 labels: ...value omitted...
 name: mysql-persistent ②
objects: ③
- apiVersion: v1
 kind: Secret
 metadata:
 annotations: ...annotations omitted...
 name: ${DATABASE_SERVICE_NAME} ④
 stringData: ...stringData omitted...
- apiVersion: v1
 kind: Service
 metadata:
 annotations: ...annotations omitted...
 name: ${DATABASE_SERVICE_NAME}
 spec: ...spec omitted...
- apiVersion: v1
 kind: PersistentVolumeClaim
 metadata:
 name: ${DATABASE_SERVICE_NAME}
 spec: ...spec omitted...
- apiVersion: v1
 kind: Deployment
 metadata:
 annotations: ...output omitted...
 name: ${DATABASE_SERVICE_NAME}
 spec: ...output omitted...
parameters: ⑤
- ...MEMORY_LIMIT parameter omitted...
- ...NAMESPACE parameter omitted...
- description: The name of the OpenShift Service exposed for the database.
 displayName: Database Service Name
 name: DATABASE_SERVICE_NAME ⑥
 required: true
 value: mysql
- ...MYSQL_USER parameter omitted...
- description: Password for the MySQL connection user.
 displayName: MySQL Connection Password
 from: '[a-zA-Z0-9]{16}' ⑦
 generate: expression
```

```
name: MYSQL_PASSWORD
required: true
- ...MYSQL_ROOT_PASSWORD parameter omitted...
- ...MYSQL_DATABASE parameter omitted...
- ...VOLUME_CAPACITY parameter omitted...
- ...MYSQL_VERSION parameter omitted...
```

- ➊ Définit une liste de balises arbitraires à associer à ce modèle. Entrez l'une de ces balises dans l'interface utilisateur pour trouver ce modèle.
- ➋ Définit le nom du modèle.
- ➌ La section `objects` définit la liste des ressources OpenShift pour ce modèle. Ce modèle crée quatre ressources : un `Secret`, un `Service`, un `PersistentVolumeClaim` et un `Deployment`.
- ➍ Les noms des quatre objets de ressources sont définis sur la valeur du paramètre `DATABASE_SERVICE_NAME`.
- ➎ La section `parameters` contient une liste de neuf paramètres.
- ➏ Les ressources de modèles définissent souvent leurs attributs en utilisant les valeurs de ces paramètres, comme le montre le paramètre `DATABASE_SERVICE_NAME`.
- ➐ Si vous ne spécifiez pas de valeur pour le paramètre `MYSQL_PASSWORD` lorsque vous créez une application avec ce modèle, OpenShift génère un mot de passe correspondant à cette expression régulière.

Vous pouvez également publier un nouveau modèle sur le cluster OpenShift, de sorte que d'autres développeurs puissent créer une application à partir du modèle.

Supposons que vous ayez une application de liste de tâches nommée `todo` qui nécessite un objet OpenShift `Deployment`, `Service` et `Route` à déployer. Vous créez un fichier de définition de modèle YAML qui définit les attributs de ces ressources OpenShift, ainsi que les définitions des paramètres requis. En supposant que le modèle est défini dans le fichier `todo-template.yaml`, utilisez la commande `oc create` pour publier le modèle d'application :

```
[user@host deploy-multicontainer]$ oc create -f todo-template.yaml
template.template.openshift.io/todonodejs-persistent created
```

Par défaut, le modèle est créé sous le projet en cours, à moins que vous n'en spécifiez un autre à l'aide de l'option `-n`, comme illustré dans l'exemple suivant :

```
[user@host deploy-multicontainer]$ oc create -f todo-template.yaml \
> -n openshift
```



### Important

Tout modèle créé sous l'espace de noms `openshift` (projet OpenShift) est disponible dans la console Web sous le `Developer Catalog` de la perspective `Developer`. De plus, tout modèle créé dans le projet actuel est accessible à partir de ce projet.

## Paramètres

Les modèles définissent un ensemble de paramètres, qui sont des valeurs affectées. Les ressources OpenShift définies dans le modèle peuvent obtenir leurs valeurs de configuration en se référant aux *paramètres nommés*. Les paramètres d'un modèle peuvent avoir des valeurs par défaut, mais elles sont facultatives. Toute valeur par défaut peut être remplacée lors du traitement du modèle.

Chaque valeur de paramètre peut être définie explicitement à l'aide de la commande `oc process`, ou être générée par OpenShift en fonction de la configuration des paramètres.

Il existe deux façons de lister les paramètres disponibles à partir d'un modèle. La première consiste à utiliser la commande `oc describe` :

```
[user@host ~]$ oc describe template mysql-persistent -n openshift
Name: mysql-persistent
Namespace: openshift
Created: 12 days ago
Labels: samplesoperator.config.openshift.io/managed=true
Description: MySQL database service, with ...description omitted...
Annotations: iconClass=icon-mysql-database
 openshift.io/display-name=MySQL
 ...output omitted...
 tags=database,mysql

Parameters:
Name: MEMORY_LIMIT
Display Name: Memory Limit
Description: Maximum amount of memory the container can use.
Required: true
Value: 512Mi

Name: NAMESPACE
Display Name: Namespace
Description: The OpenShift Namespace where the ImageStream resides.
Required: false
Value: openshift

...output omitted...

Name: MYSQL_VERSION
Display Name: Version of MySQL Image
Description: Version of MySQL image to be used (8.0, or latest).
Required: true
Value: 8.0

Object Labels: template=mysql-persistent-template

Message: ...output omitted... in your project: ${DATABASE_SERVICE_NAME}.

Username: ${MYSQL_USER}
Password: ${MYSQL_PASSWORD}
Database Name: ${MYSQL_DATABASE}
Connection URL: mysql://${DATABASE_SERVICE_NAME}:3306/
```

```
For more information about using this template, ...output omitted...
```

Objects:

|                       |                           |
|-----------------------|---------------------------|
| Secret                | \${DATABASE_SERVICE_NAME} |
| Service               | \${DATABASE_SERVICE_NAME} |
| PersistentVolumeClaim | \${DATABASE_SERVICE_NAME} |
| Deployment            | \${DATABASE_SERVICE_NAME} |

La seconde consiste à utiliser oc process avec l'option --parameters :

```
[user@host ~]$ oc process --parameters mysql-persistent -n openshift
NAME DESCRIPTION GENERATOR VALUE
MEMORY_LIMIT Maximum a...
NAMESPACE The OpenS...
DATABASE_SERVICE_NAME The name ...
MYSQL_USER Username ...
MYSQL_PASSWORD Password ...
MYSQL_ROOT_PASSWORD Password ...
MYSQL_DATABASE Name of t...
VOLUME_CAPACITY Volume sp...
MYSQL_VERSION Version o...

```

## Traitement d'un modèle à l'aide de la CLI

Lorsque vous traitez un modèle, vous générez une liste de ressources en vue de créer une application. Pour traiter un modèle, utilisez la commande oc process :

```
[user@host ~]$ oc process -f <filename>
```

La commande précédente traite un fichier de modèle, soit au format JSON ou YAML, et renvoie la liste de ressources vers la sortie standard. La liste de ressources en sortie est au format JSON. Pour générer la liste de ressources au format YAML, utilisez -o yaml avec la commande oc process :

```
[user@host ~]$ oc process -o yaml -f <filename>
```

Une autre possibilité consiste à traiter un modèle à partir du projet en cours ou du projet openshift :

```
[user@host ~]$ oc process <uploaded-template-name>
```



### Note

La commande oc process renvoie une liste de ressources vers la sortie standard. Cette sortie peut être redirigée vers un fichier :

```
[user@host ~]$ oc process -o yaml -f filename > myapp.yaml
```

**chapitre 7 |** Déploiement d'applications multiconteneurs

Les modèles génèrent souvent des ressources avec des attributs configurables basés sur les paramètres de modèle. Pour remplacer un paramètre, utilisez l'option `-p` suivie d'une paire `<name>=<value>`.

```
[user@host ~]$ oc process -o yaml -f mysql.yaml \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi > mysqlProcessed.yaml
```

Pour créer l'application, utilisez le fichier de définition de ressources YAML généré :

```
[user@host ~]$ oc create -f mysqlProcessed.yaml
```

Sinon, il est possible de traiter le modèle et de créer l'application sans enregistrer de fichier de définition de ressources, en utilisant un tube UNIX :

```
[user@host ~]$ oc process -f mysql.yaml -p MYSQL_USER=dev \
> -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi | oc create -f -
```

Pour utiliser un modèle dans le projet `openshift` afin de créer une application dans votre projet, commencez par exporter le modèle :

```
[user@host ~]$ oc get template mysql-persistent -o yaml \
> -n openshift > mysql-persistent-template.yaml
```

Ensuite, identifiez les valeurs appropriées pour les paramètres du modèle et traitez le modèle :

```
[user@host ~]$ oc process -f mysql-persistent-template.yaml \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi | oc create -f -
```

Vous pouvez également utiliser deux barres obliques (//) pour indiquer l'espace de noms comme un élément du nom du modèle :

```
[user@host ~]$ oc process openshift//mysql-persistent \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi | oc create -f -
```

Sinon, il est possible de créer une application à l'aide de la commande `oc new-app` en passant le nom du modèle comme argument de l'option `--template` :

```
[user@host ~]$ oc new-app --template=mysql-persistent \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi
```



## Références

De plus amples informations à l'attention des développeurs au sujet des modèles sont disponibles dans la section *Using Templates* de la documentation OpenShift Container Platform :

### Guide du développeur

[https://access.redhat.com/documentation/en-us/  
openshift\\_container\\_platform/4.10/html-single/images/index#using-templates](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/images/index#using-templates)

## ► Exercice guidé

# Création d'une application avec un modèle

Dans cet exercice, vous allez déployer l'application `To Do List` dans OpenShift Container Platform à l'aide d'un modèle afin de définir les ressources dont votre application a besoin pour s'exécuter.

## Résultats

Vous devez pouvoir créer et déployer une application dans OpenShift Container Platform à l'aide d'un modèle JSON fourni.

## 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-openshift start
```

## Instructions

- 1. Créez l'application `To Do List` à partir du modèle JSON fourni.

- 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 à Openshift Container Platform.

```
[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".
```

Si la commande `oc login` vous invite à utiliser des connexions non sécurisées, répondez `y` (oui).

- 1.3. Créez dans OpenShift un projet *template* à utiliser pour cet exercice. Exécutez la commande suivante pour créer le projet *template*.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-template
Now using project ...output omitted...
```

1.4. Examinez le projet template.

À l'aide de votre éditeur préféré, ouvrez et examinez le modèle situé sur `/home/student/D0180/labs/multicontainer-openshift/todo-template.json`. Notez les ressources suivantes définies dans le modèle 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`.

1.5. Traitez le modèle et créez les ressources d'application.

Utilisez la commande `oc process` pour traiter le fichier de modèle. Ce modèle requiert que l'espace de noms Quay.io récupère les images de conteneur. Utilisez la commande `pipe` pour envoyer le résultat à la commande `oc create`.

Dans la fenêtre de terminal, exécutez la commande suivante :

```
[student@workstation ~]$ cd /home/student/D0180/labs/multicontainer-openshift
[student@workstation multicontainer-openshift]$ oc process \
> -f todo-template.json \
> | oc create -f -
pod/mysql created
pod/todoapi created
service/todoapi created
service/mysql created
persistentvolumeclaim/dbclaim created
```

1.6. Examinez le déploiement.

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-openshift]$ oc get pods -w
NAME READY STATUS RESTARTS AGE
mysql 0/1 ContainerCreating 0 27s
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-openshift]$ 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 /home/student/D0180/labs/multicontainer-openshift
[student@workstation multicontainer-openshift]$ 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-openshift]$ 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-openshift]$ oc status | grep -o "http:.*com"
http://todoapi-${{RHT_OCP4_DEV_USER}}-template.${{RHT_OCP4_WILDCARD_DOMAIN}}
```

- 4.2. Utilisez `curl` afin de tester l'API REST pour l'application `To Do List`.

```
[student@workstation multicontainer-openshift]$ curl -w "\n" \
> $(oc status | grep -o "http:.*com")/todo/api/items/1
{"id":1,"description":"Pick up newspaper","done":false}
```

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. Accédez au répertoire /home/student/.

```
[student@workstation multicontainer-openshift]$ cd ~
[student@workstation ~]$
```

4.4. Ouvrez Firefox sur workstation et saisissez l'adresse `http://todoapi-  
${RHT_OCP4_DEV_USER}-template.${RHT_OCP4_WILDCARD_DOMAIN}/  
todo/` : vous devez voir l'application To Do List.

**Note**

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.

The screenshot displays the 'To Do List Application' interface. On the left, there is a table titled 'To Do List' showing two tasks:

| ID | Description    | Done  |   |
|----|----------------|-------|---|
| 1  | Pick up new... | false | X |
| 2  | Buy groceries  | true  | X |

At the bottom of this section are navigation buttons: First, Previous, 1 (highlighted in blue), Next, Last.

On the right, there is a form titled 'Add Task' with fields for 'Description' (containing 'Add Description.') and 'Completed' (with an unchecked checkbox). Below the form are 'Clear' and 'Save' buttons.

**Figure 7.6: Application To Do List**

## Fin

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

```
[student@workstation ~]$ lab multicontainer-openshift 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 `Repositories Visibility` de l'Annexe C, *Création d'un compte Quay* pour obtenir des informations sur la façon de modifier la visibilité des référentiels.

4. Accédez au répertoire `/home/student/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 ~/DO180/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>
```

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 ~/DO180/labs/multicontainer-review
```

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 `Repositories Visibility` de l'??? 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 `/home/student/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.

- 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

**chapitre 7 |** Déploiement d'applications multiconteneurs

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. Accédez au 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

# Résolution des problèmes relatifs aux applications en conteneur

### Objectif

Résoudre les problèmes relatifs à une application en conteneur déployée sur OpenShift.

### Résultats

- Résoudre les problèmes relatifs au déploiement et à la build d'une application sur OpenShift.
- Mettre en œuvre des techniques de résolution des problèmes et de débogage des applications en conteneur.

### Sections

- Résolution des problèmes relatifs aux déploiements et aux builds S2I (et exercice guidé)
- Résolution des problèmes relatifs aux applications en conteneur (et exercice guidé)

### Atelier

- Résolution des problèmes relatifs aux applications en conteneur

# Résolution des problèmes relatifs aux déploiements et aux builds S2I

## Résultats

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

- Résoudre les problèmes relatifs aux étapes de déploiement et de build d'une application sur OpenShift.
- Analyser les journaux OpenShift afin d'identifier les problèmes au cours du processus de build et de déploiement.

## Présentation du processus S2I

Le processus Source-to-Image (S2I) offre une façon simple de créer automatiquement des images basées sur le langage de programmation du code source d'application dans OpenShift. Bien que ce processus constitue souvent une façon pratique de déployer rapidement des applications, des problèmes peuvent survenir lors du processus de création d'images S2I, soit en raison des caractéristiques du langage de programmation, soit de par l'environnement d'exécution qui exige que les développeurs et les administrateurs travaillent de concert.

Il est important de bien comprendre le workflow de base de la plupart des langages de programmation pris en charge par OpenShift. Le processus de création d'images S2I se compose de deux étapes principales :

- Étape de compilation : compilation du code source, téléchargement des dépendances de bibliothèque et empaquetage de l'application sous la forme d'une image de conteneur. En outre, lors de cette étape, l'image est envoyée au registre OpenShift en vue de l'étape de déploiement. Les ressources OpenShift `BuildConfig` (BC) dirigent l'étape de compilation.
- Étape de déploiement : démarrage d'un pod et mise à disposition de l'application pour OpenShift. Cette étape s'exécute après l'étape de compilation, mais uniquement si cette dernière s'est déroulée correctement. Les ressources OpenShift `Deployment` pilotent l'étape de déploiement.

Pour le processus S2I, chaque application utilise ses propres objets `BuildConfig` et `Deployment`, dont le nom correspond à celui de l'application. En cas d'échec de la compilation, le processus de déploiement est abandonné.

Le processus S2I démarre chaque étape dans un pod distinct. Le processus de build crée un pod nommé `<application-name>-build-<number>-<string>`. À chaque tentative de compilation, l'étape s'exécute intégralement et enregistre un journal. Une fois la compilation réussie, l'application démarre sur un pod distinct appelé `<application-name>-<string>`.

La console Web OpenShift peut être utilisée pour accéder aux détails de chaque étape. Pour identifier un problème de compilation, il est possible d'évaluer et d'analyser les journaux d'une compilation en cliquant sur le lien `Builds` situé dans le panneau gauche, comme illustré ci-dessous.

| Name              | Namespace       | Status     | Created       |
|-------------------|-----------------|------------|---------------|
| B httpd-example-1 | NS troubleshoot | ✓ Complete | 2 minutes ago |

Figure 8.1: Compiler des instances d'un projet

À chaque tentative de build, un historique est fourni, balisé avec un numéro, pour évaluation.  
Cliquez sur le nom de la compilation pour accéder à la page des détails de celle-ci :

Figure 8.2: Vue détaillée d'une instance de compilation

À chaque tentative de build, un historique est fourni, balisé avec un numéro, pour évaluation.  
Cliquez sur le nom de la compilation pour accéder à la page des détails de celle-ci.

L'onglet **Logs** de la page des détails de la compilation affiche la sortie générée par l'exécution de la compilation. Ces journaux sont pratiques pour identifier les problèmes de compilation.

Utilisez le lien **Deployment** sous la section **Workloads** dans le panneau de gauche pour identifier les problèmes lors de l'étape de déploiement.

Après avoir sélectionné l'objet de déploiement approprié, les détails apparaissent dans la section **Details**.

L'interface de ligne de commande `oc` comporte plusieurs sous-commandes permettant de gérer les journaux. L'interface Web comprend également un ensemble de commandes qui fournissent des informations sur chaque étape. Par exemple, pour récupérer les journaux d'une configuration de build, exéutez la commande suivante :

```
$ oc logs bc/<application-name>
```

Si une compilation échoue, après avoir trouvé et résolu les problèmes, exécutez la commande suivante pour demander une nouvelle compilation :

```
$ oc start-build <application-name>
```

En émettant cette commande, OpenShift génère automatiquement un nouveau pod avec le processus de compilation.

Les journaux de déploiement peuvent être vérifiés avec la commande `oc` :

```
$ oc logs deployment/<application-name>
```

Si le déploiement est en cours d'exécution ou a échoué, la commande renvoie les journaux du processus de déploiement de processus. Sinon, la commande renvoie les journaux du pod de l'application.

## Description de problèmes courants

Parfois, le code source nécessite une personnalisation qui peut ne pas être disponible dans les environnements en conteneur, tels que les informations d'identification de la base de données, l'accès au système de fichiers ou les informations de la file d'attente de messages. Ces valeurs prennent généralement la forme de variables d'environnement internes. Les développeurs utilisant le processus S2I peuvent avoir besoin d'accéder à ces informations.

La commande `oc logs` fournit des informations importantes sur les processus de compilation, de déploiement et d'exécution d'une application dans le cadre de l'exécution d'un pod. Les journaux peuvent indiquer les valeurs ou options manquantes qui doivent être activées, les paramètres ou les indicateurs non valides ou encore les incompatibilités au niveau de l'environnement.



### Note

Les journaux d'application doivent être clairement étiquetés afin d'identifier rapidement les problèmes, sans qu'il soit nécessaire de connaître le fonctionnement interne du conteneur.

## Résolution de problèmes d'autorisation courants

OpenShift exécute des conteneurs S2I en utilisant Red Hat Enterprise Linux comme image de base, et toute différence d'exécution peut entraîner l'échec du processus S2I. Parfois, les développeurs sont confrontés à des problèmes d'autorisation, tels qu'un accès refusé en raison de mauvaises autorisations ou des autorisations d'environnement incorrectes définies par les administrateurs. Les images S2I imposent qu'un utilisateur autre que `root` soit choisi pour l'accès aux systèmes de fichiers et aux ressources externes. En outre, Red Hat Enterprise Linux 8 applique des politiques SELinux qui limitent l'accès à certaines ressources du système de fichiers, à certains ports réseau ou à certains processus.

Pour certains conteneurs, un ID d'utilisateur spécifique peut s'avérer nécessaire. Cependant, S2I est conçu pour exécuter des conteneurs à l'aide d'un utilisateur aléatoire, conformément à la politique de sécurité OpenShift par défaut.

## chapitre 8 | Résolution des problèmes relatifs aux applications en conteneur

Le fichier Dockerfile suivant crée un conteneur Nexus. Notez l'instruction USER indiquant que l'utilisateur nexus doit être utilisé :

```
FROM ubi8/ubi:8.1
...contents omitted...
RUN chown -R nexus:nexus ${NEXUS_HOME}

USER nexus
WORKDIR ${NEXUS_HOME}

VOLUME ["/opt/nexus/sonatype-work"]
...contents omitted...
```

Essayer d'utiliser l'image générée par ce Dockerfile sans prendre en compte les lecteurs d'autorisations de volume entraîne des erreurs lors du démarrage du conteneur :

```
$ oc logs nexus-1-wzjrn
...output omitted...
... org.sonatype.nexus.util.LockFile - Failed to write lock file
...FileNotFoundException: /opt/nexus/sonatype-work/nexus.lock (Permission denied)
...output omitted...
... org.sonatype.nexus.webapp.WebappBootstrap - Failed to initialize
...lStateException: Nexus work directory already in use: /opt/nexus/sonatype-work
...output omitted...
```

Pour remédier à ce problème, vous pouvez assouplir la sécurité du projet OpenShift à l'aide de la commande `oc adm policy` :

```
[user@host ~]$ oc adm policy add-scc-to-user anyuid -z default
```

Cette commande `oc adm policy` permet à OpenShift d'exécuter des processus de conteneur avec des utilisateurs non root. Mais les systèmes de fichiers utilisés dans le conteneur doivent également être disponibles pour l'utilisateur en cours d'exécution. Ceci est particulièrement important lorsque le conteneur contient des montages de volume.

Pour éviter les problèmes liés aux autorisations du système de fichiers, les dossiers locaux utilisés pour les montages de volume de conteneur doivent répondre aux exigences suivantes :

- L'utilisateur qui exécute les processus de conteneur doit être le propriétaire du dossier ou posséder les droits nécessaires. Utilisez la commande `chown` pour mettre à jour la propriété du dossier.
- Le dossier local doit satisfaire aux exigences de SELinux pour pouvoir être utilisé en tant que volume de conteneur. Attribuez le groupe `container_file_t` au dossier en utilisant la commande `semanage fcontext -a -t container_file_t <folder>`, puis actualisez les autorisations avec la commande `restorecon -R <folder>`.

## Résolution des problèmes liés aux paramètres non valides

Les applications à plusieurs conteneurs peuvent partager des paramètres, tels que des informations d'identification. Vérifiez que les mêmes valeurs atteignent tous les conteneurs de l'application. Par exemple, pour une application Python qui s'exécute dans un conteneur, connectée à un autre conteneur exécutant une base de données, assurez-vous que les deux conteneurs utilisent le même nom d'utilisateur et le même mot de passe pour la base de données.

## chapitre 8 | Résolution des problèmes relatifs aux applications en conteneur

En règle générale, les journaux du pod d'application permettent d'identifier clairement les problèmes et la façon de les résoudre.

Une bonne pratique pour centraliser les paramètres partagés consiste à les stocker dans **ConfigMaps**. Ces **ConfigMaps** peuvent être injectés viaDeployment dans des conteneurs en tant que variables d'environnement. Injecter le même **ConfigMap** dans des conteneurs différents garantit que non seulement les mêmes variables d'environnement sont disponibles, mais également les mêmes valeurs. Voir la définition de ressource de pod suivante :

```
apiVersion: v1
kind: Pod
...output omitted...
spec:
 containers:
 - name: test-container
 ...output omitted...
 env:
 - name: ENV_1 ❶
 valueFrom:
 configMapKeyRef:
 name: configMap_name1
 key: configMap_key_1
 ...output omitted...
 envFrom:
 - configMapRef:
 name: configMap_name_2 ❷
 ...output omitted...
```

- ❶ Une variable d'environnement ENV\_1 est injectée dans le conteneur. Sa valeur est la valeur pour l'entrée configMap\_key\_1 dans le configMap configMap\_name1.
- ❷ Toutes les entrées dans configMap\_name\_2 sont injectées dans le conteneur en tant que variables d'environnement avec le même nom et les mêmes valeurs.

## Résolution des erreurs de montage de volume

Lors du redéploiement d'une application qui utilise un volume persistant sur un système de fichiers local, il se peut qu'un pod ne parvienne pas à allouer une revendication de volume persistant, même si le volume persistant indique que la revendication en question est libérée.

Pour remédier à ce problème, supprimez la revendication de volume persistant, puis le volume persistant. Ensuite, recréez le volume persistant :

```
oc delete pv <pv_name>
oc create -f <pv_resource_file>
```

## Résolution des problèmes liés aux images obsolètes

OpenShift extrait des images de la source indiquée dans un flux d'images, à moins qu'il ne trouve une image mise en mémoire cache locale sur le nœud où il est prévu que le pod s'exécute. Si vous envoyez au registre une nouvelle image portant le même nom et la même balise, vous devez la supprimer de chaque nœud sur lequel le pod est planifié à l'aide de la commande `podman rmi`.

Exécutez la commande `oc adm prune` pour découvrir une méthode automatisée pour la suppression d'images obsolètes et d'autres ressources.



## Références

Vous trouverez des informations complémentaires sur la résolution des problèmes liés aux images dans la section *Images* de la documentation OpenShift Container Platform accessible à l'adresse suivante :

### Création d'images

[https://docs.openshift.com/container-platform/4.10/openshift\\_images/create-images.html](https://docs.openshift.com/container-platform/4.10/openshift_images/create-images.html)

La documentation sur la façon d'utiliser ConfigMap pour créer des variables d'environnement de conteneur est disponible dans *Consuming in Environment Variables* sur le

### Configuration d'un pod pour utiliser ConfigMaps

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-pod-configmap/#define-container-environment-variables-using-configmap-data>

## ► Exercice guidé

# Résolution des problèmes relatifs à une compilation OpenShift

Dans cet exercice, vous allez résoudre les problèmes relatifs à un processus de build et de déploiement OpenShift.

## Résultats

Vous devriez pouvoir identifier et résoudre les problèmes qui se produisent lors du processus de compilation et de déploiement d'une application Node.js.

## 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.

Récupérez les fichiers d'atelier, et vérifiez que Docker et le cluster OpenShift sont en cours d'exécution en exécutant la commande suivante.

```
[student@workstation ~]$ lab troubleshoot-s2i start
```

## Instructions

- 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
```

- 2. 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...
```

- 3. 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 troubleshoot-s2i
Switched to a new branch 'troubleshoot-s2i'
[student@workstation D0180-apps]$ git push -u origin troubleshoot-s2i
...output omitted...
* [new branch] troubleshoot-s2i -> s2i
Branch 'troubleshoot-s2i' set up to track remote branch 'troubleshoot-s2i' from
'origin'.
```

- 4. Connectez-vous à OpenShift à l'aide de l'utilisateur, du mot de passe et de l'URL d'API maîtresse configurés.

```
[student@workstation D0180-apps]$ oc login -u "${RHT_OCP4_DEV_USER}" \
> -p "${RHT_OCP4_DEV_PASSWORD}"
Login successful.
...output omitted...
```

Créez un nouveau projet nommé *youruser-nodejs*.

```
[student@workstation D0180-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-nodejs
Now using project "youruser-nodejs" on server "https://
api.cluster.lab.example.com"
...output omitted...
```

- 5. Compilez une nouvelle application Node.js au moyen de l'image Hello World située à l'adresse <https://github.com/yourgituser/D0180-apps/> dans le répertoire nodejs-helloworld.

- 5.1. Exécutez la commande `oc new-app` pour créer l'application Node.js. La commande est fournie dans le fichier `~/D0180/labs/troubleshoot-s2i/command.txt`.

```
[student@workstation D0180-apps]$ oc new-app \
> --context-dir=nodejs-helloworld \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#troubleshoot-s2i \
> -i nodejs:16-ubi8 --name nodejs-hello --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs
--> Found image e9b0166 ...output omitted...

Node.js 16
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "nodejs-hello" created
buildconfig.build.openshift.io "nodejs-hello" created
deployment.apps "nodejs-hello" created
service "nodejs-hello" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/nodejs-hello' 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/nodejs-hello'
Run 'oc status' to view your app.
```

L'option `-i` indique l'image de compilateur à utiliser ; `nodejs:12` dans ce cas.

L'option `--context-dir` définit le dossier à l'intérieur du projet qui contient le code source de l'application à compiler.

L'option `--build-env` définit une variable d'environnement pour le pod du compilateur. Dans ce cas, elle fournit la variable d'environnement `npm_config_registry` au pod du compilateur, afin qu'elle puisse atteindre le registre NPM.



### Important

Dans la commande précédente, il ne doit pas y avoir d'espace entre `registry=` et l'URL du serveur Nexus.

- 5.2. Attendez que l'application termine le processus de build en surveillant la progression à l'aide de la commande `oc get pods -w`. Le pod passe du statut `running` à `Error` :

```
[student@workstation D0180-apps]$ oc get pods -w
NAME READY STATUS RESTARTS AGE
nodejs-hello-1-build 1/1 Running 0 15s
nodejs-hello-1-build 0/1 Error 0 73s
^C
```

Le processus de build échoue et aucune application n'est donc en cours d'exécution. Les échecs de build sont généralement dus à des erreurs de syntaxe dans le code source ou à des dépendances manquantes. L'étape suivante recherche les causes de cette erreur.

- 5.3. Examinez les erreurs qui se sont produites lors du processus de build.

La build est déclenchée par la configuration correspondante (`bc`), créée par OpenShift au démarrage du processus S2I. Par défaut, le processus OpenShift S2I crée une configuration de compilation nommée `nodejs-hello`, responsable du déclenchement du processus de compilation.

Exécutez la commande `oc` avec la sous-commande `logs` dans une fenêtre de terminal pour confirmer le résultat du processus de compilation :

```
[student@workstation D0180-apps]$ oc logs bc/nodejs-hello
Cloning "https://github.com/yourgituser/D0180-apps" ...
Commit: f7cd8963ef353d9173c3a21dcccf402f3616840b (Initial commit...
...output omitted...
STEP 8: RUN /usr/libexec/s2i/assemble
--> Installing application source ...
--> Installing all dependencies
npm ERR! code ETARGET
npm ERR! notarget No matching version found for express@~4.14.2.
npm ERR! notarget In most cases you or one of your dependencies are requesting
npm ERR! notarget a package version that doesn't exist.
npm ERR! notarget
npm ERR! notarget It was specified as a dependency of 'src'
npm ERR! notarget

npm ERR! A complete log of this run can be found in:
npm ERR! /opt/app-root/src/.npm/_logs/2019-10-25T12_37_56_853Z-debug.log
`error: build error: error building at STEP "RUN /usr/libexec/s2i/assemble": error
while running runtime: exit status 1
`...output omitted...
```

Le journal indique qu'une erreur s'est produite lors du processus de compilation. Cette sortie indique qu'il n'existe pas de version compatible pour la dépendance

express. Mais la raison en est que le format utilisé par la dépendance express n'est pas valide.

► 6. Mettez à jour le processus de build pour le projet.

Chaque développeur utilise une version non standard de la structure Express disponible en local sur sa station de travail. En raison des normes de l'entreprise, la version doit être téléchargée à partir du registre officiel de Node.js ; à partir de l'entrée du développeur, elle est compatible avec la version 4.14.x.

6.1. Réparez le fichier `package.json`.

Utilisez l'éditeur de votre choix pour ouvrir le fichier `~/D0180-apps/nodejs-helloworld/package.json`. Examinez les versions des dépendances fournies par les développeurs. Une version incorrecte est utilisée pour la dépendance Express. Celle-ci est incompatible avec la version prise en charge fournie par l'entreprise (~4.14.2). Mettez à jour la version de la dépendance comme suit.

```
{
 "name": "nodejs-helloworld",
 ...output omitted...
 "dependencies": {
 "express": "4.14.x"
 }
}
```



**Note**

Remarquez le x dans la version. Cela indique que la version la plus récente doit être utilisée, mais que la version doit commencer par 4.14..

6.2. Validez et envoyez les modifications apportées au projet.

Dans la fenêtre de terminal, exécutez la commande suivante pour valider et envoyer les modifications :

```
[student@workstation D0180-apps]$ git commit -am "Fixed Express release"
...output omitted...
1 file changed, 1 insertion(+), 1 deletion(-)
[student@workstation D0180-apps]$ git push
...output omitted...
To https://github.com/yourgituser/D0180-apps
 ef6557d..73a82cd troubleshoot-s2i -> troubleshoot-s2i
```

► 7. Relancez le processus S2I.

7.1. Pour redémarrer l'étape de compilation, exécutez la commande suivante :

```
[student@workstation D0180-apps]$ oc start-build bc/nodejs-hello
build.build.openshift.io/nodejs-hello-2 started
```

L'étape de build est redémarrée et un nouveau pod de build est créé. Vérifiez le journal en exécutant la commande `oc logs`.

```
[student@workstation D0180-apps]$ oc logs -f bc/nodejs-hello
Cloning "https://github.com/yougituser/D0180-apps" ...
Commit: ea2125c1bf4681dd9b79ddf920d8d8be38cf3b (Fixed Express release)
...output omitted...
Pushing image ...image-registry.svc:5000/nodejs/nodejs-hello:latest...
...output omitted...
Push successful
```

Le processus de build a réussi, mais cela n'indique pas que l'application est démarrée.

- 7.2. Évaluez l'état du processus de build en cours. Exécutez la commande `oc get pods` pour vérifier l'état de l'application Node.js.

```
[student@workstation D0180-apps]$ oc get pods
```

Selon la sortie suivante, la deuxième build est terminée mais l'application est en état d'erreur.

| NAME                          | READY | STATUS                  | RESTARTS | AGE   |
|-------------------------------|-------|-------------------------|----------|-------|
| nodejs-hello-1-build          | 0/1   | Error                   | 0        | 7m59s |
| <b>nodejs-hello-2-build</b>   | 0/1   | <b>Completed</b>        | 0        | 54s   |
| nodejs-hello-7b99966464-fw4r8 | 0/1   | <b>CrashLoopBackOff</b> | 1        | 18s   |

Le nom du pod d'application (`nodejs-hello-7b99966464-fw4r8`) est généré de manière aléatoire et peut différer du vôtre.

- 7.3. Passez en revue les journaux générés par le pod d'application.

```
[student@workstation D0180-apps]$ oc logs nodejs-hello-7b99966464-fw4r8
...output omitted...
npm info using npm@8.1.2
npm info using node@v16.13.1
npm ERR! missing script: start
...output omitted...
```



### Note

La commande `oc logs nodejs-hello-7b99966464-fw4r8` vide les journaux du pod de déploiement. En cas de déploiement réussi, cette commande vide les journaux du pod d'applications, comme indiqué précédemment.

Si les journaux de déploiement n'affichent pas l'erreur, vérifiez les journaux du pod d'application en exécutant la commande `oc logs POD` en remplaçant `POD` par le nom du pod qui a le statut `CrashLoopBackOff`.

L'application ne parvient pas à démarrer car la déclaration de script de démarrage est manquante.

- 8. Corrigez le problème en mettant à jour le code d'application.

- 8.1. Mettez à jour le fichier `package.json` pour définir une commande de démarrage.

Le résultat précédent indique que, dans le fichier ~/D0180-apps/nodejs-helloworld/package.json, l'attribut `start` manque dans le champ `scripts`. L'attribut `start` définit une commande devant s'exécuter au démarrage de l'application. Il appelle le binaire `node`, qui exécute l'application `app.js`.

Pour remédier au problème, ajoutez l'attribut suivant au fichier `package.json`. N'oubliez pas la virgule après le crochet.

```
...
 "description": "Hello World!",
 "main": "app.js",
 "scripts": { "start": "node app.js" },
 "author": "Red Hat Training",
...
```

#### 8.2. Validez et envoyez les modifications apportées au projet :

```
[student@workstation D0180-apps]$ git commit -am "Added start up script"
...output omitted...
1 file changed, 3 insertions(+)
[student@workstation D0180-apps]$ git push
...output omitted...
To https://github.com/yourgituser/D0180-apps
 73a82cd..a5a0411 troubleshoot-s2i -> troubleshoot-s2i
```

Continuez l'étape de déploiement à partir du processus S2I.

#### 8.3. Redémarrez l'étape de build.

```
[student@workstation D0180-apps]$ oc start-build bc/nodejs-hello
build.build.openshift.io/nodejs-hello-3 started
```

#### 8.4. Évaluez l'état du processus de build en cours. Exécutez la commande pour récupérer l'état de l'application Node.js. Attendez la fin de la dernière build.

```
[student@workstation D0180-apps]$ oc get pods -w
NAME READY STATUS RESTARTS AGE
nodejs-hello-1-build 0/1 Error 0 26m
nodejs-hello-2-build 0/1 Completed 0 19m
nodejs-hello-3-build 1/1 Running 0 9s
nodejs-hello-7b99966464-fw4r8 0/1 CrashLoopBackOff 8 19m
nodejs-hello-6dc88b7b7-dvtcz 0/1 Pending 0 0s
nodejs-hello-6dc88b7b7-dvtcz 0/1 Pending 0 0s
nodejs-hello-6dc88b7b7-dvtcz 0/1 ContainerCreating 0 0s
nodejs-hello-3-build 0/1 Completed 0 37s
nodejs-hello-6dc88b7b7-dvtcz 0/1 ContainerCreating 0 2s
nodejs-hello-6dc88b7b7-dvtcz 1/1 Running 0 3s
nodejs-hello-7b99966464-fw4r8 0/1 Terminating 8 19m
nodejs-hello-7b99966464-fw4r8 0/1 Terminating 8 19m
nodejs-hello-7b99966464-fw4r8 0/1 Terminating 8 19m
```

Selon la sortie, la compilation a réussi et l'application est en mesure de démarrer sans erreur. La sortie fournit également un aperçu de la manière dont le pod de déploiement (`nodejs-hello-3-build`) a été créé, et indique qu'il s'est exécuté

**chapitre 8 |** Résolution des problèmes relatifs aux applications en conteneur

correctement et s'est terminé. Étant donné que le nouveau pod d'application est disponible (`nodejs-hello-7b99966464-dvtcz`), l'ancien (`nodejs-hello-7b99966464-fw4r8`) est déployé.

- 8.5. Passez en revue les journaux générés par le pod d'application `nodejs-hello`.

```
[student@workstation D0180-apps]$ oc logs nodejs-hello-6dc88b7b7-dvtcz
Environment:
 DEV_MODE=false
 NODE_ENV=production
 DEBUG_PORT=5858
Launching via npm...
npm info it worked if it ends with ok
npm info using npm@8.1.2
npm info using node@v16.13.1
npm info lifecycle nodejs-helloworld@1.0.0~prestart: nodejs-helloworld@1.0.0
npm info lifecycle nodejs-helloworld@1.0.0~start: nodejs-helloworld@1.0.0

> nodejs-helloworld@1.0.0 start
> node app.js

Example app listening on port 8080!
```

L'application s'exécute maintenant sur le port 8080.

► 9. Testez l'application.

- 9.1. Exécutez la commande `oc` avec la sous-commande `expose` pour exposer l'application :

```
[student@workstation D0180-apps]$ oc expose svc/nodejs-hello
route.route.openshift.io/nodejs-hello exposed
```

- 9.2. Récupérez l'adresse associée à l'application.

```
[student@workstation D0180-apps]$ oc get route -o yaml
apiVersion: v1
items:
- apiVersion: route.openshift.io/v1
 kind: Route
...output omitted...
spec:
 host: nodejs-hello-${RHT_OCP4_DEV_USER}-nodejs.${RHT_OCP4_WILDCARD_DOMAIN}
 port:
 targetPort: 8080-tcp
 to:
 kind: Service
 name: nodejs-hello
...output omitted...
```

- 9.3. Accédez à l'application à partir de la machine virtuelle `workstation` en utilisant la commande `curl`:

```
[student@workstation DO180-apps]$ curl -w "\n" \
> http://nodejs-hello-${RHT_OCP4_DEV_USER}-nodejs.${RHT_OCP4_WILDCARD_DOMAIN}
Hello world!
```

La sortie montre que l'application est en cours d'exécution.

## Fin

Sur `workstation`, exéutez le script `lab troubleshoot-s2i finish` pour mettre fin à cet atelier.

```
[student@workstation DO180-apps]$ lab troubleshoot-s2i finish
```

Cela met fin à cet exercice.

# Résolution des problèmes relatifs aux applications en conteneur

## Résultats

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

- Mettre en œuvre des techniques de résolution des problèmes et de débogage des applications en conteneur.
- Utiliser la fonction de réacheminement de ports de l'outil client OpenShift.
- Afficher les journaux de conteneur.
- Afficher les événements de cluster OpenShift.

## Réacheminement de ports pour la résolution de problèmes

Parfois, les développeurs et les administrateurs système ont besoin d'un accès réseau spécial à un conteneur ; contrairement aux utilisateurs de l'application. Par exemple, les développeurs peuvent avoir besoin d'utiliser la console d'administration pour une base de données ou un service de messagerie, ou les administrateurs système peuvent utiliser l'accès SSH à un conteneur pour redémarrer un service terminé. De tels accès réseau, sous la forme de ports réseau, ne sont généralement pas exposés aux configurations de conteneur par défaut et requièrent généralement des clients spécialisés utilisés par des développeurs et des administrateurs système.

Podman fournit des fonctions de réacheminement de port en utilisant l'option `-p` en même temps que la sous-commande `run`. Dans ce cas, il n'y a pas de différence entre l'accès réseau pour un accès standard à l'application et pour la résolution des problèmes. Pour mémoire, voici un exemple de configuration d'une mise en correspondance du réacheminement du port depuis l'hôte vers un serveur de base de données exécuté à l'intérieur d'un conteneur :

```
$ podman run --name db -p 30306:3306 mysql
```

La commande précédente met en correspondance le port hôte 30306 avec le port 3306 sur le conteneur `db`. Ce conteneur est créé à partir de l'image `mysql`. Celle-ci lance un serveur MySQL qui écoute sur le port 3306.

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.

Voici un exemple de la commande `oc port-forward` :

```
$ oc port-forward db 30306 3306
```

La commande précédente réachemine le port 30306 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.

Alors que la méthode `podman run -p` de mise en correspondance (réacheminement de ports) ne peut être configurée que lorsque le conteneur est démarré, la mise en correspondance avec la commande `oc port-forward` peut être créée et éliminée à tout moment après la création d'un pod.



### Note

Créer un service de type `NodePort` pour un pod de base de données revient à exécuter `podman run -p`. Toutefois, Red Hat déconseille l'utilisation de l'approche `NodePort` pour éviter l'exposition du service aux connexions directes. La mise en correspondance avec le réacheminement de ports dans OpenShift est considérée comme une option plus sûre.

## Activation du débogage distant avec réacheminement de ports

La fonction de réacheminement de ports peut également être utilisée pour activer le débogage à distance. De nombreux environnements de développement intégré (IDE) offrent la possibilité de déboguer une application à distance.

Red Hat CodeReady Studio, par exemple, permet d'utiliser le protocole JDWP (Java Debug Wire Protocol) pour établir une communication entre son débogueur et la machine virtuelle Java (JVM). Lorsque cette communication est active, les développeurs peuvent parcourir chaque ligne de code exécutée en temps réel.

Pour que le protocole JDWP fonctionne, la machine virtuelle Java qui héberge l'application doit être démarrée avec les options qui activent le débogage à distance. Par exemple, les utilisateurs WildFly et JBoss EAP doivent configurer ces options au démarrage du serveur d'applications. La ligne suivante du fichier `standalone.conf` active le débogage à distance en ouvrant le port TCP JDWP 8787 pour une instance WildFly ou EAP exécutée en mode autonome :

```
JAVA_OPTS="$JAVA_OPTS \
> -agentlib:jdwp=transport=dt_socket,address=8787,server=y,suspend=n"
```

Lorsque le serveur démarre et que le débogueur écoute sur le port 8787, une correspondance du réacheminement de ports doit être créée pour réacheminer les connexions depuis un port TCP inutilisé local vers le port 8787 du pod EAP. Si aucune machine virtuelle Java avec débogage à distance ne s'exécute sur la station de travail developer, le port local peut également être 8787.

## chapitre 8 | Résolution des problèmes relatifs aux applications en conteneur

La commande suivante suppose l'utilisation d'un pod WildFly nommé **jappserver**, qui exécute un conteneur à partir d'une image configurée précédemment pour activer le débogage à distance :

```
$ oc port-forward jappserver 8787:8787
```

Une fois que le débogueur est activé et que la mise en correspondance du réacheminement de ports est créée, les utilisateurs peuvent définir des points d'arrêt dans l'environnement de développement intégré (IDE) de leur choix et exécuter le débogueur en plaçant le pointeur de la souris sur le nom d'hôte et le port de débogage (dans ce cas, 8787).

## Accès aux journaux de conteneur

Podman et OpenShift offrent la possibilité d'afficher des journaux dans des conteneurs et des pods en cours d'exécution afin de faciliter la résolution des problèmes. Cependant, aucun n'a connaissance des journaux spécifiques de l'application. Tous deux s'attendent à ce que l'application soit configurée de manière à envoyer tout le résultat de journalisation à la sortie standard.

Du point de vue du système d'exploitation hôte, un conteneur est simplement une arborescence de processus. Lorsque Podman démarre un conteneur, que ce soit directement ou sur le cluster RHOC, il redirige l'erreur standard et la sortie standard du conteneur, en les enregistrant sur disque dans le cadre du stockage éphémère du conteneur. De cette manière, les journaux de conteneur peuvent être visionnés à l'aide des commandes `podman` et `oc` (même après l'arrêt du conteneur), mais ils ne peuvent pas être supprimés.

Pour récupérer la sortie d'un conteneur en cours d'exécution, utilisez la commande `podman` suivante :

```
$ podman logs <containerName>
```

Dans OpenShift, la commande suivante renvoie la sortie d'un conteneur à l'intérieur d'un pod :

```
$ oc logs <podName> [-c <containerName>]
```



### Note

S'il y a un seul conteneur, son nom est facultatif, étant donné que `oc` est défini par défaut sur le seul conteneur en cours d'exécution et renvoie la sortie.

## Événements OpenShift

Certains développeurs considèrent que les journaux Podman et OpenShift sont d'un niveau trop bas, ce qui complique la résolution des problèmes. Heureusement, OpenShift fournit une fonction de journalisation et d'audit de haut niveau, connue sous le nom d'événements.

Les événements OpenShift signalent les opérations importantes, comme le démarrage d'un conteneur ou l'élimination d'un pod.

Pour lire les événements OpenShift, utilisez la sous-commande `get` avec le type de ressource `events` pour la commande `oc`, comme suit :

```
$ oc get events
```

Les événements listés de cette façon par la commande `oc` ne sont pas filtrés et couvrent l'ensemble du cluster RHOC. L'utilisation d'un caractère pipe pour des filtres UNIX standard, tels que `grep`, peut s'avérer utile, mais OpenShift offre une autre solution pour consulter les événements de cluster. Cette approche est fournie par la sous-commande `describe`.

Par exemple, pour récupérer uniquement les événements liés à un pod `mysql`, reportez-vous au champ `Events` dans la sortie de la commande `oc describe pod mysql`.

```
$ oc describe pod mysql
...output omitted...
Events:
FirstSeen LastSeen Count From Reason Message
Wed, 10 ... Wed, 10 ... 1 {scheduler} scheduled Successfully as...
...output omitted...
```

## Accès aux conteneurs en cours d'exécution

Les commandes `podman logs` et `oc logs` peuvent être utiles pour visualiser la sortie envoyée par n'importe quel conteneur. Toutefois, la sortie n'affiche pas nécessairement toutes les informations disponibles si l'application est configurée de manière à envoyer les journaux à un fichier. Dans d'autres scénarios de dépannage, il peut être nécessaire d'examiner l'environnement du conteneur, tel qu'il apparaît pour les processus à l'intérieur du conteneur, tel que la vérification de la connectivité externe.

En guise de solution, Podman et OpenShift fournissent la sous-commande `exec`, qui permet de créer des processus au sein d'un conteneur actif, avec la sortie et l'entrée standard de ces processus redirigées vers le terminal utilisateur. L'écran suivant affiche l'utilisation de la commande `podman exec` :

```
$ podman exec [options] container command [arguments]
```

La syntaxe générale de la commande `oc exec` est la suivante :

```
$ oc exec [options] pod [-c container] -- command [arguments]
```

Pour exécuter une seule commande interactive ou démarrer un shell, ajoutez les options `-it`. L'exemple suivant démarre un shell Bash pour le pod `myhttpd` :

```
$ oc exec -it myhttpd /bin/bash
```

Vous pouvez vous servir de cette commande pour accéder aux journaux d'application enregistrés sur disque (dans le cadre du stockage éphémère du conteneur). Par exemple, pour afficher le journal d'erreurs Apache à partir d'un conteneur, exéutez la commande suivante :

```
$ podman exec apache-container cat /var/log/httpd/error_log
```

## Remplacement des binaires du conteneur

De nombreuses images de conteneur n'offrent pas toutes les commandes de dépannage que les utilisateurs s'attendent à trouver dans des installations de système d'exploitation standard, telles que `telnet`, `netcat`, `ip` ou `traceroute`. La suppression de l'image des utilitaires de base ou des fichiers binaires permet à l'image de rester mince, et d'exécuter ainsi plusieurs conteneurs par hôte.

Pour accéder temporairement à certaines commandes manquantes, une technique consiste à monter les dossiers de binaires de l'hôte, comme `/bin`, `/sbin` et `/lib`, en tant que volumes à l'intérieur du conteneur. Cela est possible, car l'option `-v` de la commande `podman run` n'exige pas la présence d'instructions `VOLUME` correspondantes dans le `Containerfile` de l'image de conteneur.



### Note

Pour accéder à ces commandes dans OpenShift, vous devez modifier la définition du ressource de pod afin de pouvoir définir les objets `volumeMounts` et `volumeClaims`. Vous avez également besoin de créer un volume persistant `hostPath`.

La commande suivante démarre un conteneur et remplace le dossier `/bin` de l'image par celui de l'hôte. Elle lance également un shell interactif à l'intérieur de ce conteneur :

```
$ podman run -it -v /bin:/bin image /bin/bash
```



### Note

Le répertoire de binaires à remplacer dépend de l'image du système d'exploitation de base. Certaines commandes, par exemple, ont besoin des bibliothèques partagées du répertoire `/lib`. Certaines distributions Linux présentent des contenus différents dans `/bin`, `/usr/bin`, `/lib` ou `/usr/lib`, qui nécessiteraient d'utiliser l'option `-v` pour chaque répertoire.

Vous pouvez également inclure ces utilitaires dans l'image de base. Pour ce faire, ajoutez des instructions dans une définition de build `Containerfile`. Par exemple, examinez l'extrait suivant d'une définition `Containerfile`, qui est l'enfant de l'image `rhel7.5`. L'instruction `RUN` installe les outils couramment utilisés pour la résolution des problèmes liés au réseau :

```
FROM ubi7/ubi:7.7

RUN yum install -y \
 less \
 dig \
 ping \
 iputils && \
 yum clean all
```

Lorsque l'image est compilée et le conteneur créé, elle est identique à une image de conteneur `rhel7.5`, plus les outils supplémentaires disponibles.

## Transfert de fichiers vers et depuis des conteneurs

Lors du dépannage ou de la gestion d'une application, il peut s'avérer nécessaire d'extraire ou de transférer des fichiers vers ou à partir de conteneurs en cours d'exécution ; des fichiers journaux ou des fichiers de configuration, par exemple. Il existe plusieurs façons de déplacer des fichiers dans et hors de conteneurs, comme décrit dans la liste suivante.

### Montages de volumes

Pour copier des fichiers de l'hôte sur un conteneur, une autre méthode consiste à utiliser des montages de volumes. Vous pouvez monter un répertoire local pour copier des données dans un conteneur. La commande suivante, par exemple, définit le répertoire d'hôte `/conf` comme volume à utiliser pour le répertoire de configuration Apache dans le conteneur. Vous disposez ainsi d'une méthode pratique pour gérer le serveur Apache sans devoir régénérer l'image de conteneur.

```
$ podman run -v /conf:/etc/httpd/conf -d do180/apache
```

### podman cp

La sous-commande `cp` permet aux utilisateurs d'extraire et de copier des fichiers dans un conteneur actif. Pour copier un fichier dans un conteneur nommé `todoapi`, exéutez la commande suivante.

```
$ podman cp standalone.conf todoapi:/opt/jboss/standalone/conf/standalone.conf
```

Pour copier un fichier du conteneur vers l'hôte, inversez l'ordre de la commande précédente.

```
$ podman cp todoapi:/opt/jboss/standalone/conf/standalone.conf .
```

La commande `podman cp` présente l'avantage de fonctionner avec des conteneurs qui ont déjà été démarrés, tandis que la solution suivante (montages de volumes) exige que des modifications soient apportées à la commande utilisée pour démarrer un conteneur.

### podman exec

Dans le cas des conteneurs déjà actifs, la commande `podman exec` peut être « canalisée » afin de transmettre des fichiers (en entrée et en sortie) en ajoutant les commandes exécutées dans le conteneur. L'exemple ci-dessous montre comment transmettre et exécuter un fichier SQL à l'intérieur d'un conteneur MySQL :

```
$ podman exec -i <container> mysql -uroot -proot < /path/on/host/db.sql
```

En appliquant le même concept, il est possible de récupérer des données d'un conteneur actif et de les placer dans la machine hôte. Un exemple utile est l'utilisation de l'utilitaire `mysqldump`, qui crée une sauvegarde de la base de données MySQL à partir du conteneur et la place sur l'hôte.

```
$ podman exec -it <containerName> sh \
> -c 'exec mysqldump -h"$MYSQL_PORT_3306_TCP_ADDR" \
> -P"$MYSQL_PORT_3306_TCP_PORT" \
> -uroot -p"$MYSQL_ENV_MYSQL_ROOT_PASSWORD" items' > db_dump.sql
```

La commande précédente utilise les variables d'environnement du conteneur pour se connecter au serveur MySQL afin d'exécuter la commande `mysqldump`. Elle redirige la sortie vers un fichier sur la machine hôte. Elle suppose que l'image de conteneur fournit l'utilitaire

`mysqldump`, de sorte qu'il n'est pas nécessaire d'installer les outils d'administration MySQL sur l'hôte.

La commande `oc rsync` offre des fonctionnalités semblables à `podman cp` pour les conteneurs exécutés sous des pods OpenShift.



### Références

Vous trouverez des informations complémentaires sur le réacheminement de ports dans la section *Port Forwarding* de la documentation OpenShift Container Platform sur

#### 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 plus d'informations sur les commandes CLI relatives au réacheminement de ports dans le chapitre *Port Forwarding* de la documentation OpenShift Container Platform sur

#### Développement d'applications

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.10/html-single/building\\_applications/index](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/building_applications/index)

## ► Exercice guidé

# Configuration des journaux de conteneur Apache pour le débogage

Dans cet exercice, vous allez configurer un conteneur httpd Apache afin d'envoyer les journaux à `stdout`, puis passer en revue les événements et journaux Podman.

## Résultats

Vous serez en mesure de configurer un conteneur httpd Apache afin d'envoyer des journaux de débogage à `stdout`, puis de les consulter à l'aide de la commande `podman logs`.

## 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.

Récupérez les fichiers d'atelier, et vérifiez que Docker et le cluster OpenShift sont en cours d'exécution en exécutant la commande suivante.

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

## Instructions

- ▶ 1. Configurez un serveur Web Apache pour envoyer des messages de journal à la sortie standard et mettre à jour le niveau de journalisation par défaut.
  - 1.1. Le niveau de journalisation par défaut pour l'image Apache httpd est `warn`. Définissez le niveau de journalisation par défaut pour le conteneur sur `debug`, et redirigez les messages de journaux vers `stdout`, en remplaçant le fichier de configuration `httpd.conf` par défaut. Pour ce faire, créez une image personnalisée de la machine virtuelle `workstation`.Passez rapidement en revue le fichier `httpd.conf` personnalisé situé à l'adresse `/home/student/D0180/labs/troubleshoot-container/conf/httpd.conf`.
    - Observez la directive `ErrorLog` dans le fichier :

```
ErrorLog "/dev/stdout"
```

La directive envoie les messages du journal d'erreurs httpd vers la sortie standard du conteneur.

- Observez la directive `LogLevel` dans le fichier.

```
LogLevel debug
```

Cette directive définit le niveau de journalisation par défaut sur `debug`.

- Observez la directive `CustomLog` dans le fichier.

```
CustomLog "/dev/stdout" common
```

La directive redirige les messages du journal d'accès httpd vers la sortie standard du conteneur.

► **2.** Créez un conteneur personnalisé pour y enregistrer le fichier de configuration mis à jour.

- 2.1. Dans la fenêtre de terminal, exécutez les commandes suivantes pour créer une nouvelle image.

```
[student@workstation ~]$ cd ~/DO180/labs/troubleshoot-container
[student@workstation troubleshoot-container]$ podman build \
> -t troubleshoot-container .
STEP 1: FROM quay.io/redhattraining/httpd-parent
...output omitted...
STEP 5: COMMIT troubleshoot-container
e23d...c1de
[student@workstation troubleshoot-container]$ cd ~
```

- 2.2. Vérifiez que l'image a bien été créée.

```
[student@workstation ~]$ podman images
```

La nouvelle image doit être disponible dans le stockage local.

| REPOSITORY                          | TAG    | IMAGE ID | CREATED       | SIZE  |
|-------------------------------------|--------|----------|---------------|-------|
| localhost/troubleshoot-container    | latest | e23df... | 9 seconds ago | 137MB |
| quay.io/redhattraining/httpd-parent | latest | 0eba3... | 4 weeks ago   | 137MB |

► **3.** Créez un conteneur httpd à partir de l'image personnalisée.

```
[student@workstation ~]$ podman run \
> --name troubleshoot-container -d \
> -p 10080:80 troubleshoot-container
4c8bb12815cc02f4eef0254632b7179bd5ce230d83373b49761b1ac41fc067a9
```

► **4.** Passez en revue les événements et messages du journal du conteneur.

- 4.1. Consultez les messages du journal de débogage en provenance du conteneur à l'aide de la commande `podman logs` :

```
[student@workstation ~]$ podman logs -f troubleshoot-container
... [mpm_event:notice] [pid 1:tid...] AH00489: Apache/2.4.37 ...
... [mpm_event:info] [pid 1:tid...] AH00490: Server built: Apr 5 2019 07:31:21
... [core:notice] [pid 1:tid...] AH00094: Command line: 'httpd -D FOREGROUND'
... [core:debug] [pid 1:tid ...]: AH02639: Using SO_REUSEPORT: yes (1)
... [mpm_event:debug] [pid 6:tid ...]: AH02471: start_threads: Using epoll
... [mpm_event:debug] [pid 7:tid ...]: AH02471: start_threads: Using epoll
... [mpm_event:debug] [pid 8:tid ...]: AH02471: start_threads: Using epoll
```

Notez les journaux de débogage, disponibles dans la sortie standard.

- 4.2. Ouvrez un nouveau terminal et accédez à la page d'accueil du serveur Web en utilisant la commande `curl`:

```
[student@workstation ~]$ curl http://127.0.0.1:10080
Hello from the httpd-parent container!
```

- 4.3. Vérifiez les nouvelles entrées dans le journal. Regardez dans le terminal qui exécute la commande `podman logs` pour voir les nouvelles entrées.

```
...output omitted...
10.0.2.2 - - [31/Mar/2021:14:06:03 +0000] "GET / HTTP/1.1" 200 39
```

- 4.4. Arrêtez la commande Podman avec `Ctrl+C`.

## Fin

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

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

L'exercice guidé est maintenant terminé.

## ► Open Lab

# Résolution des problèmes relatifs aux applications en conteneur

### Résultats

Vous devriez pouvoir identifier et résoudre les problèmes qui se produisent lors du processus de compilation et de déploiement d'une application Node.js.

### 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 troubleshoot-review start
```

### Instructions

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 :  

```
source ./vars.sh
```
2. 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 :  

```
git clone https://github.com/red-hat-labs/D0180-apps.git
cd D0180-apps
git checkout master
```
3. Créez une branche nommée **troubleshoot-review** et poussez-la pour enregistrer toutes les modifications que vous avez effectuées au cours de cet exercice :  

```
git branch -b troubleshoot-review
git push origin troubleshoot-review
```
4. Connectez-vous à OpenShift à l'aide de l'utilisateur, du mot de passe et de l'URL d'API maîtresse configurés.
5. Créez un projet nommé **youruser-nodejs-app** :
6. Dans le projet **youruser-nodejs-app**, créez une application nommée **nodejs-dev**. Utilisez le flux d'images **nodejs:12** et le code source dans le répertoire **nodejs-app**, situé dans la branche du référentiel Git **troubleshoot-review** que vous avez créée lors d'une étape précédente. Définissez la valeur de la variable d'environnement **npm\_config\_registry** avec **http://\${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs**.

Attendez-vous à ce que le processus de compilation de l'application échoue. Surveillez le processus de compilation et identifiez l'échec de la compilation.

7. Mettez à jour la version de la dépendance **express** dans le fichier **package.json** avec la valeur **4.x**. Validez et poussez les modifications vers le référentiel Git.
8. Compilez à nouveau l'application. Vérifiez que l'application est compilée sans erreur.
9. Vérifiez que l'application n'est pas en cours d'exécution à cause d'une erreur d'exécution. Consultez les journaux et identifiez le problème.

10. Le journal indique que le fichier `server.js` tente de charger un module nommé `http-error`, qui n'est pas disponible. Remplacez le module requis dans la première ligne du `server.js` par `html-errors`, qui est un module valide dans le fichier `packages`. Validez et transmettez les modifications apportées à l'application au référentiel Git. Compilez à nouveau l'application. Une fois l'application compilée, vérifiez qu'elle est en cours d'exécution.
11. Créez une route pour l'application et testez l'accès à l'application. Attendez-vous à un message d'erreur. Passez en revue les journaux afin d'identifier l'erreur.
12. Remplacez `process.environment` par `process.env` dans le fichier `server.js` pour corriger l'erreur. Validez et transmettez les modifications apportées à l'application au référentiel Git. Compilez à nouveau l'application. Lorsque la nouvelle application se déploie, vérifiez qu'elle ne génère pas d'erreur lorsque vous accédez à l'URL de l'application.

## Évaluation

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

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

## Fin

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

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

L'atelier est maintenant terminé.

## ► Solution

# Résolution des problèmes relatifs aux applications en conteneur

### Résultats

Vous devriez pouvoir identifier et résoudre les problèmes qui se produisent lors du processus de compilation et de déploiement d'une application Node.js.

### 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 troubleshoot-review start
```

### Instructions

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
```

2. 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...
```

3. Créez une branche nommée **troubleshoot-review** et poussez-la pour enregistrer toutes les modifications que vous avez effectuées au cours de cet exercice :

```
[student@workstation D0180-apps]$ git checkout -b troubleshoot-review
Switched to a new branch 'troubleshoot-review'
[student@workstation D0180-apps]$ git push -u origin troubleshoot-review
...output omitted...
* [new branch] troubleshoot-review -> troubleshoot-review
Branch 'troubleshoot-review' set up to track remote branch 'troubleshoot-review'
from 'origin'.
```

4. Connectez-vous à OpenShift à l'aide de l'utilisateur, du mot de passe et de l'URL d'API maîtresse configurés.

```
[student@workstation D0180-apps]$ oc login -u "${RHT_OCP4_DEV_USER}" \
> -p "${RHT_OCP4_DEV_PASSWORD}" "${RHT_OCP4_MASTER_API}"
Login successful.
...output omitted...
```

5. Créez un projet nommé *youruser-nodejs-app*:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-nodejs-app
Now using project "youruser-nodejs-app" on server "https://
api.cluster.lab.example.com"
...output omitted...
```

6. Dans le projet *youruser-nodejs-app*, créez une application nommée *nodejs-dev*. Utilisez le flux d'images *nodejs:12* et le code source dans le répertoire *nodejs-app*, situé dans la branche du référentiel *Git troubleshoot-review* que vous avez créée lors d'une étape précédente. Définissez la valeur de la variable d'environnement *npm\_config\_registry* avec *http://\${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs*.

Attendez-vous à ce que le processus de compilation de l'application échoue. Surveillez le processus de compilation et identifiez l'échec de la compilation.

6.1. Exécutez la commande *oc new-app* pour créer l'application Node.js.

```
[student@workstation ~]$ oc new-app --name nodejs-dev \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#troubleshoot-review \
> -i nodejs:16-ubi8 --context-dir=nodejs-app --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs
--> Found image a2b5ec2 ...output omitted...

Node.js 16
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "nodejs-dev" created
buildconfig.build.openshift.io "nodejs-dev" created
deployment.apps "nodejs-dev" created
service "nodejs-dev" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/nodejs-dev'` 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 service/nodejs-dev'
Run '+oc status' to view your app.
```

6.2. Suivez l'état d'avancement de la compilation avec la commande *oc logs -f bc/nodejs-dev*:

```
[student@workstation ~]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
STEP 8: RUN /usr/libexec/s2i/assemble
--> Installing application source ...
```

```
--> Installing all dependencies
npm ERR! code ETARGET
npm ERR! notarget No matching version found for express@4.20.
npm ERR! notarget In most cases you or one of your dependencies are requesting
npm ERR! notarget a package version that doesn't exist.
npm ERR! notarget
npm ERR! notarget It was specified as a dependency of 'src'
npm ERR! notarget

npm ERR! A complete log of this run can be found in:
npm ERR! /opt/app-root/src/.npm/_logs/2019-10-28T11_30_27_657Z-debug.log
subprocess exited with status 1
subprocess exited with status 1
error: build error: error building at STEP "RUN /usr/libexec/s2i/assemble": exit
status 1
```

Le processus de build échoue et aucune application n'est donc en cours d'exécution. Le journal de compilation indique qu'il n'existe pas de version du paquetage `express` qui corresponde à la spécification de version `4.20.x`.

- 6.3. Utilisez la commande `oc get pods` pour confirmer que l'application n'est pas déployée :

```
[student@workstation ~]$ oc get pods
NAME READY STATUS RESTARTS AGE
nodejs-dev-1-build 0/1 Error 0 2m
```

7. Mettez à jour la version de la dépendance `express` dans le fichier `package.json` avec la valeur `4.x`. Validez et poussez les modifications vers le référentiel Git.
  - 7.1. Modifiez le fichier `package.json` dans le sous-répertoire `nodejs-app` et remplacez la version de la dépendance `express` par `4.x`. Enregistrez le fichier.

```
[student@workstation DO180-apps]$ cd nodejs-app
```

```
[student@workstation nodejs-app]$ sed -i s/4.20/4.x/ package.json
```

Le fichier contient les éléments suivants :

```
[student@workstation nodejs-app]$ cat package.json
{
 "name": "nodejs-app",
 "version": "1.0.0",
 "description": "Hello World App",
 "main": "server.js",
 "author": "Red Hat Training",
 "license": "ASL",
 "dependencies": {
 "express": "4.x",
 "html-errors": "latest"
 }
}
```

**chapitre 8 |** Résolution des problèmes relatifs aux applications en conteneur

- 7.2. Validez et envoyez les modifications apportées au projet.

Dans la fenêtre de terminal, exécutez la commande suivante pour valider et envoyer les modifications :

```
[student@workstation nodejs-app]$ git commit -am "Fixed Express release"
...output omitted...
1 file changed, 1 insertion(+), 1 deletion(-)
[student@workstation nodejs-app]$ git push
...output omitted...
To https://github.com/yourgituser/D0180-apps/
 ef6557d..73a82cd troubleshoot-review -> troubleshoot-review
```

8. Compilez à nouveau l'application. Vérifiez que l'application est compilée sans erreur.

- 8.1. Utilisez la commande `oc start-build` pour compiler à nouveau l'application.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-2 started
```

- 8.2. Utilisez la commande `oc logs` pour surveiller les journaux du processus de compilation :

```
[student@workstation nodejs-app]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
Pushing image ...image-registry.svc:5000/nodejs-app/nodejs-dev:latest ...
...output omitted...
Push successful
```

La compilation réussit si une image est transmise au registre OpenShift interne.

9. Vérifiez que l'application n'est pas en cours d'exécution à cause d'une erreur d'exécution. Consultez les journaux et identifiez le problème.

- 9.1. Utilisez la commande `oc get pods` pour vérifier le statut de déploiement du pod d'application. Finalement, vous voyez que le premier déploiement d'application a le statut `CrashLoopBackoff`.

| NAME                | READY | STATUS           | RESTARTS | AGE   |
|---------------------|-------|------------------|----------|-------|
| nodejs-dev-1-build  | 0/1   | Error            | 0        | 4m27s |
| nodejs-dev-1-deploy | 0/1   | Completed        | 0        | 28s   |
| nodejs-dev-1-skf56  | 0/1   | CrashLoopBackOff | 2        | 25s   |
| nodejs-dev-2-build  | 0/1   | Completed        | 0        | 58s   |

- 9.2. Utilisez la commande `oc logs -f deployment/nodejs-dev` pour suivre les journaux pour le déploiement de l'application :

```
[student@workstation nodejs-app]$ oc logs -f deployment/nodejs-dev
Environment:
 DEV_MODE=false
 NODE_ENV=production
```

```
DEBUG_PORT=5858
...output omitted...

Error: Cannot find module 'http-error'

...output omitted...

npm info nodejs-app@1.0.0 Failed to exec start script
...output omitted...
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! nodejs-app@1.0.0 start: node `server.js`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the nodejs-app@1.0.0 start script.
npm ERR! This is probably not a problem with npm. There is likely additional
logging output above.
npm timing npm Completed in 159ms
...output omitted...
```

Le journal indique que le fichier `server.js` tente de charger un module nommé `http-error`. La variable `dependencies` dans le fichier `packages` indique que le nom du module est `html-errors`, et non `http-error`.

10. Le journal indique que le fichier `server.js` tente de charger un module nommé `http-error`, qui n'est pas disponible. Remplacez le module requis dans la première ligne du `server.js` par `html-errors`, qui est un module valide dans le fichier `packages`. Validez et transmettez les modifications apportées à l'application au référentiel Git. Compilez à nouveau l'application. Une fois l'application compilée, vérifiez qu'elle est en cours d'exécution.
  - 10.1. Corrigez l'orthographe du module dans la première ligne de `server.js` de `http-error` à `html-errors`. Enregistrez le fichier.

```
[student@workstation nodejs-app]$ sed -i s/http-error/html-errors/ server.js
```

Le fichier contient les éléments suivants :

```
[student@workstation nodejs-app]$ cat server.js
var createError = require('html-errors');

var express = require('express');
app = express();

app.get('/', function (req, res) {
 res.send('Hello World from pod: ' + process.env.HOSTNAME + '\n')
});

app.listen(8080, function () {
 console.log('Example app listening on port 8080!');
});
```

- 10.2. Validez et envoyez les modifications apportées au projet.

```
[student@workstation nodejs-app]$ git commit -am "Fixed module typo"
...output omitted...
1 file changed, 1 insertion(+), 1 deletion(-)
[student@workstation nodejs-app]$ git push
...output omitted...
To https://github.com/yourgituser/D0180-apps/
 ef6557d..73a82cd troubleshoot-review -> troubleshoot-review
```

10.3. Utilisez la commande `oc start-build` pour compiler à nouveau l'application.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-3 started
```

10.4. Utilisez la commande `oc logs` pour surveiller les journaux du processus de compilation :

```
[student@workstation nodejs-app]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
Pushing imageimage-registry.svc:5000/nodejs-app/nodejs-dev:latest ...
...output omitted...
Push successful
```

10.5. Utilisez la commande `oc get pods -w` pour surveiller le déploiement de pods pour l'application `nodejs-dev` :

```
[student@workstation nodejs-app]$ oc get pods -w
NAME READY STATUS RESTARTS AGE
nodejs-dev-1-build 0/1 Error 0 11m
nodejs-dev-1-deploy 0/1 Completed 0 7m11s
nodejs-dev-2-9nds4 1/1 Running 0 56s
nodejs-dev-2-build 0/1 Completed 0 7m41s
nodejs-dev-2-deploy 0/1 Completed 0 61s
nodejs-dev-3-build 0/1 Completed 0 94s
```

Après une troisième compilation, le deuxième déploiement affiche le statut `Running`.

11. Créez une route pour l'application et testez l'accès à l'application. Attendez-vous à un message d'erreur. Passez en revue les journaux afin d'identifier l'erreur.

11.1. Utilisez la commande `oc expose` pour créer une route pour l'application `nodejs-dev` :

```
[student@workstation nodejs-app]$ oc expose svc nodejs-dev
route.route.openshift.io/nodejs-dev exposed
```

11.2. Utilisez la commande `oc get route` pour récupérer l'URL de la route `nodejs-dev` :

```
[student@workstation nodejs-app]$ oc get route
NAME HOST/PORT
nodejs-dev nodejs-dev-your_user-nodejs-app.wildcard_domain ...
```

**chapitre 8 |** Résolution des problèmes relatifs aux applications en conteneur

- 11.3. Utilisez la commande `curl` pour accéder à la route. Attendez-vous à ce qu'un message d'erreur s'affiche.

```
[student@workstation nodejs-app]$ curl \
> nodejs-dev-${RHT_OCP4_DEV_USER}-nodejs-app.${RHT_OCP4_WILDCARD_DOMAIN}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Internal Server Error</pre>
</body>
</html>
```

- 11.4. Passez en revue les journaux pour le déploiement `nodejs-dev`:

```
[student@workstation nodejs-app]$ oc logs -f deployment/nodejs-dev
Environment:
 DEV_MODE=false
 NODE_ENV=production
 DEBUG_PORT=5858
Launching via npm...
npm info it worked if it ends with ok
npm info using npm@6.14.8
npm info using node@v12.19.1
npm info lifecycle nodejs-app@1.0.0~prestart: nodejs-app@1.0.0
npm info lifecycle nodejs-app@1.0.0~start: nodejs-app@1.0.0

Example app listening on port 8080!
TypeError: Cannot read property 'HOSTNAME' of undefined
...output omitted...
```

La section correspondante du fichier `server.js` est :

```
app.get('/', function (req, res) {
 res.send('Hello World from pod: ' + process.environment.HOSTNAME + '\n')
});
```

Un objet `process` dans Node.js contient une référence à un objet `env`, et non à un objet `environment`.

12. Remplacez `process.environment` par `process.env` dans le fichier `server.js` pour corriger l'erreur. Validez et transmettez les modifications apportées à l'application au référentiel Git. Compilez à nouveau l'application. Lorsque la nouvelle application se déploie, vérifiez qu'elle ne génère pas d'erreur lorsque vous accédez à l'URL de l'application.
- 12.1. Remplacez `process.environment` par `process.env` dans le fichier `server.js` pour corriger l'erreur.

```
[student@workstation nodejs-app]$ sed -i \
> s/process.environment/process.env/ server.js
```

Le fichier contient les éléments suivants :

```
[student@workstation nodejs-app]$ cat server.js
var createError = require('html-errors');

var express = require('express');
app = express();

app.get('/', function (req, res) {
 res.send('Hello World from pod: ' + process.env.HOSTNAME + '\n')
});

app.listen(8080, function () {
 console.log('Example app listening on port 8080!');
});
```

12.2. Validez et envoyez les modifications apportées au projet.

```
[student@workstation nodejs-app]$ git commit -am "Fixed process.env"
...output omitted...
1 file changed, 1 insertion(+), 1 deletion(-)
[student@workstation nodejs-app]$ git push
...output omitted...
To https://github.com/yourgituser/D0180-apps/
 ef6557d..73a82cd troubleshoot-review -> troubleshoot-review
```

12.3. Utilisez la commande `oc start-build` pour compiler à nouveau l'application.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-4 started
```

12.4. Utilisez la commande `oc logs` pour surveiller les journaux du processus de compilation :

```
[student@workstation nodejs-app]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
Pushing image ...image-registry.svc:5000/nodejs-app/nodejs-dev:latest ...
...output omitted...
Push successful
```

12.5. Utilisez la commande `oc get pods` pour surveiller le déploiement de pods pour l'application `nodejs-dev` :

```
[student@workstation nodejs-app]$ oc get pods
NAME READY STATUS RESTARTS AGE
nodejs-dev-1-build 0/1 Error 0 21m
nodejs-dev-1-deploy 0/1 Completed 0 17m
nodejs-dev-2-build 0/1 Completed 0 17m
nodejs-dev-2-deploy 0/1 Completed 0 11m
nodejs-dev-3-build 0/1 Completed 0 11m
```

**chapitre 8 |** Résolution des problèmes relatifs aux applications en conteneur

```
nodejs-dev-3-deploy 0/1 Completed 0 20s
nodejs-dev-3-wlpps 1/1 Running 0 17s
nodejs-dev-4-build 0/1 Completed 0 48s
```

Après une quatrième compilation, le troisième déploiement affiche le statut **Running**.

- 12.6. Utilisez la commande `curl` pour tester l'application. L'application affiche un message `Hello World` contenant le nom d'hôte du pod d'application :

```
[student@workstation nodejs-app]$ curl \
> nodejs-dev-${RHT_OCP4_DEV_USER}-nodejs-app.${RHT_OCP4_WILDCARD_DOMAIN}
Hello World from pod: nodejs-dev-3-wlpps
```

## Évaluation

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

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

## Fin

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

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

L'atelier est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- Les applications enregistrent généralement les activités, telles que les événements, les avertissements et les erreurs, pour faciliter l'analyse du comportement de l'application.
- Les applications conteneur doivent imprimer les données de journal sur la sortie standard, plutôt que dans un fichier, pour faciliter l'accès aux journaux.
- Pour consulter les journaux d'un conteneur déployé localement avec Podman, utilisez la commande `podman logs`.
- Utilisez la commande `oc logs` pour accéder aux journaux des objets `BuildConfig` et `Deployment`, ainsi qu'aux pods individuels au sein d'un projet OpenShift.
- L'option `-f` vous permet de surveiller la sortie du journal quasiment en temps réel à la fois pour les commandes `podman logs` et `oc logs`.
- Utilisez la commande `oc port-forward` pour vous connecter directement à un port sur un pod d'application. Utilisez uniquement cette technique sur des pods qui ne sont pas utilisés dans le cadre de la production, car les interactions peuvent modifier le comportement du pod.



## chapitre 9

# Révision complète

### Objectif

Tâches de révision depuis *Red Hat OpenShift I : conteneurs & Kubernetes*

### Résultats

- Tâches de révision depuis *Red Hat OpenShift I : conteneurs & Kubernetes*

### Sections

- Révision complète

### Atelier

- Révision complète du cours *Introduction to Containers, Kubernetes, and Red Hat OpenShift*

# Révision complète

---

## Résultats

Après avoir terminé cette section, vous devriez avoir révisé et rafraîchi les connaissances et les compétences acquises dans *Red Hat OpenShift I : conteneurs & Kubernetes*.

## Révision Red Hat OpenShift I : conteneurs & Kubernetes

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, Cr éation de services en conteneur**

Déployer un serveur en utilisant la technologie des conteneurs.

- Créer 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 créer une image de conteneur personnalisée.

## chapitre 9 | Révision complète

- Décrire les approches de création des images de conteneur personnalisées.
- Créer une image de conteneur à l'aide des commandes Containerfile courantes.

## **Chapitre 6, Déploiement 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.
- Créer des ressources Kubernetes standard.
- Créer une route vers un service.
- Construire une application au moyen de la fonction Source-to-Image d'OpenShift Container Platform.
- Créer une application à l'aide de la console Web OpenShift.

## **Chapitre 7, Déploiement 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.
- Déployer une application multiconteneur sur OpenShift.
- Déployer une application sur OpenShift à l'aide d'un modèle.

## **Chapitre 8, Résolution des problèmes relatifs aux applications en conteneur**

Résoudre les problèmes relatifs à une application en conteneur déployée sur OpenShift.

- Résoudre les problèmes relatifs au déploiement et à la build d'une application sur OpenShift.
- Mettre en œuvre des techniques de résolution des problèmes et de débogage des applications en conteneur.

## ► Open Lab

# Mise en conteneur et déploiement d'une application logicielle

Dans le cadre de cette révision, vous allez mettre en conteneur un serveur Nexus, procéder aux opérations de compilation et de test à l'aide de Podman, et le déployer sur un cluster OpenShift.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Écrire un Containerfile qui met correctement en conteneur un serveur Nexus.
- Compiler une image de conteneur de serveur Nexus et la déployer à l'aide de Podman.
- Déployer l'image de conteneur de serveur Nexus sur un cluster 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.

Exécuter le script d'installation pour cette révision approfondie.

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

Les fichiers d'atelier se trouvent dans le répertoire `/home/student/D0180/labs/comprehensive-review`. Les fichiers de solution se trouvent dans le répertoire `/home/student/D0180/solutions/comprehensive-review`.

## Instructions

Utilisez les étapes suivantes pour créer et tester un serveur Nexus en conteneur à la fois localement et dans OpenShift :

1. Créez une image de conteneur qui démarre une instance d'un serveur Nexus :
  - Le répertoire `/home/student/D0180/labs/comprehensive-review/image` contient des fichiers pour la compilation de l'image du conteneur. Exécutez le script `get-nexus-bundle.sh` pour récupérer les fichiers du serveur Nexus.
  - Écrivez un Containerfile qui met correctement en conteneur le serveur Nexus. Le Containerfile doit se trouver dans le répertoire `/home/student/D0180/labs/comprehensive-review/image`. Le Containerfile doit également :
    - Utiliser une image de base de `ubi8/ubi:8.5` et définir un chargé de maintenance arbitraire.
    - Définir la variable ARG `NEXUS_VERSION` sur `2.14.3-02`, et définir la variable d'environnement `NEXUS_HOME` sur `/opt/nexus`.

- Installer le paquetage `java-1.8.0-openjdk-devel`.
  - Exécutez une commande pour créer un utilisateur et un groupe `nexus`. Tous deux ont l'UID et le GID 1001. Modifiez les autorisations du répertoire  `${NEXUS_HOME} /` et définissez-les sur 775.
  - Décompresser le fichier `nexus-2.14.3-02-bundle.tar.gz` dans le répertoire  `${NEXUS_HOME} /`. Ajouter le fichier `nexus-start.sh` au même répertoire.

Exécutez la commande `ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2` pour créer un lien symbolique dans le conteneur. Exécutez une commande pour modifier de manière récursive la propriété du répertoire personnel `Nexus` sur `nexus:nexus`.
  - Faites en sorte que le conteneur s'exécute en tant qu'utilisateur `nexus` et définissez le répertoire de travail sur `/opt/nexus`.
  - Définissez un point de montage de volume pour le répertoire de conteneur `/opt/nexus/sonatype-work`. Le serveur `Nexus` stocke les données dans ce répertoire.
  - Définissez la commande de conteneur par défaut sur `nexus-start.sh`.

Le répertoire d'atelier `*.snippet` contient deux fichiers `/home/student/D0180/labs/comprehensive-review/image` qui fournissent les commandes nécessaires pour créer le compte `nexus` et installer Java. Utilisez les fichiers pour vous aider à écrire le Containerfile.

    - Compilez l'image du conteneur avec le nom `nexus`.
2. Compilez et testez l'image de conteneur à l'aide de Podman avec un montage de volume :
- Utilisez le script `/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh` pour démarrer un nouveau conteneur avec un montage de volume.
  - Consultez les journaux du conteneur pour vérifier que le serveur est démarré et en cours d'exécution.
  - Testez l'accès au service de conteneur à l'aide de l'URL : `http://127.0.0.1:18081/nexus`.
  - Supprimez le conteneur de test.
3. Déployez l'image de conteneur de serveur `Nexus` sur le cluster OpenShift. Vous devez effectuer les actions suivantes :
- Ajoutez la balise `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest` à l'image de conteneur du serveur `Nexus` et envoyez-la (par push) au référentiel public sur `quay.io`.
  - Créez un projet OpenShift avec le nom suivant :  `${RHT_OCP4_DEV_USER} - review`.
  - Modifiez le fichier `deploy/openshift/resources/nexus-deployment.yaml` et remplacez `RHT_OCP4_QUAY_USER` par votre nom d'utilisateur Quay. Créez les ressources Kubernetes
  - Créez une route pour le service `Nexus`. Vérifiez que vous pouvez accéder à `http://nexus- ${RHT_OCP4_DEV_USER} - review. ${RHT_OCP4_WILDCARD_DOMAIN} /nexus/` à partir de `workstation`.

## Évaluation

Après avoir déployé l'image de conteneur du serveur Nexus sur le cluster OpenShift, vérifiez votre travail en exécutant le script de notation de l'atelier :

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

## Fin

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

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

L'atelier est maintenant terminé.

## ► Solution

# Mise en conteneur et déploiement d'une application logicielle

Dans le cadre de cette révision, vous allez mettre en conteneur un serveur Nexus, procéder aux opérations de compilation et de test à l'aide de Podman, et le déployer sur un cluster OpenShift.

## Résultats

Vous serez en mesure d'effectuer les opérations suivantes :

- Écrire un Containerfile qui met correctement en conteneur un serveur Nexus.
- Compiler une image de conteneur de serveur Nexus et la déployer à l'aide de Podman.
- Déployer l'image de conteneur de serveur Nexus sur un cluster 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.

Exécuter le script d'installation pour cette révision approfondie.

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

Les fichiers d'atelier se trouvent dans le répertoire `/home/student/D0180/labs/comprehensive-review`. Les fichiers de solution se trouvent dans le répertoire `/home/student/D0180/solutions/comprehensive-review`.

## Instructions

Utilisez les étapes suivantes pour créer et tester un serveur Nexus en conteneur à la fois localement et dans OpenShift :

1. Créez une image de conteneur qui démarre une instance d'un serveur Nexus :
  - Le répertoire `/home/student/D0180/labs/comprehensive-review/image` contient des fichiers pour la compilation de l'image du conteneur. Exécutez le script `get-nexus-bundle.sh` pour récupérer les fichiers du serveur Nexus.
  - Écrivez un Containerfile qui met correctement en conteneur le serveur Nexus. Le Containerfile doit se trouver dans le répertoire `/home/student/D0180/labs/comprehensive-review/image`. Le Containerfile doit également :
    - Utiliser une image de base de `ubi8/ubi:8.5` et définir un chargé de maintenance arbitraire.
    - Définir la variable ARG `NEXUS_VERSION` sur `2.14.3-02`, et définir la variable d'environnement `NEXUS_HOME` sur `/opt/nexus`.

**chapitre 9 |** Révision complète

- Installer le paquetage `java-1.8.0-openjdk-devel`.
- Exécutez une commande pour créer un utilisateur et un groupe `nexus`. Tous deux ont l'UID et le GID 1001. Modifiez les autorisations du répertoire  `${NEXUS_HOME} /` et définissez-les sur 775.
- Décompresser le fichier `nexus-2.14.3-02-bundle.tar.gz` dans le répertoire  `${NEXUS_HOME} /`. Ajouter le fichier `nexus-start.sh` au même répertoire.

Exécutez la commande `ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2` pour créer un lien symbolique dans le conteneur. Exécutez une commande pour modifier de manière récursive la propriété du répertoire personnel Nexus sur `nexus:nexus`.

- Faites en sorte que le conteneur s'exécute en tant qu'utilisateur `nexus` et définissez le répertoire de travail sur `/opt/nexus`.
- Définissez un point de montage de volume pour le répertoire de conteneur `/opt/nexus/sonatype-work`. Le serveur Nexus stocke les données dans ce répertoire.
- Définissez la commande de conteneur par défaut sur `nexus-start.sh`.

Le répertoire d'atelier `*.snippet` contient deux fichiers `/home/student/D0180/labs/comprehensive-review/image` qui fournissent les commandes nécessaires pour créer le compte `nexus` et installer Java. Utilisez les fichiers pour vous aider à écrire le Containerfile.

- Compilez l'image du conteneur avec le nom `nexus`.

- 1.1. Exécutez le script `get-nexus-bundle.sh` pour récupérer les fichiers du serveur Nexus.

```
[student@workstation ~]$ cd /home/student/D0180/labs/comprehensive-review/image
[student@workstation image]$./get-nexus-bundle.sh
#####
100.0%
Nexus bundle download successful
```

- 1.2. Écrivez un Containerfile qui met correctement en conteneur le serveur Nexus. Accédez au répertoire `/home/student/D0180/labs/comprehensive-review/image` et créez le Containerfile.

- Indiquez l'image de base à utiliser :

```
FROM ubi8/ubi:8.5
```

- Saisissez un nom et une adresse électronique arbitraire en tant que chargé de maintenance :

```
FROM ubi8/ubi:8.5
MAINTAINER username <username@example.com>
```

- Définissez un argument de compilation pour `NEXUS_VERSION` et une variable d'environnement pour `NEXUS_HOME` :

**chapitre 9 |** Révision complète

```
FROM ubi8/ubi:8.5
MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus
```

- Installez le paquetage `java-1.8.0-openjdk-devel` à l'aide de la commande `yum`.

```
FROM ubi8/ubi:8.5
MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus

RUN yum install -y --setopt=tsflags=nodocs java-1.8.0-openjdk-devel && \
 yum clean all -y
```

- Créez le groupe et compte de service et le répertoire de base du serveur. Faites en sorte que le répertoire personnel appartienne au compte de service et modifiez les autorisations en les définissant sur 775.

```
RUN groupadd -r nexus -f -g 1001 && \
useradd -u 1001 -r -g nexus -m -d ${NEXUS_HOME} -s /sbin/nologin \
-c "Nexus User" nexus && \
chown -R nexus:nexus ${NEXUS_HOME} && \
chmod -R 755 ${NEXUS_HOME}
```

- Faites en sorte que le conteneur s'exécute en tant qu'utilisateur `nexus`.

```
USER nexus
```

- Installez le logiciel serveur Nexus dans `NEXUS_HOME` et ajoutez le script de démarrage. Notez que la directive `ADD` extraîtra les fichiers Nexus.

```
ADD nexus-${NEXUS_VERSION}-bundle.tar.gz ${NEXUS_HOME}
ADD nexus-start.sh ${NEXUS_HOME}/
```

- Créez le lien symbolique `nexus2` qui pointe vers le répertoire du serveur Nexus. Modifiez de manière récursive la propriété du répertoire `${NEXUS_HOME}` sur `nexus:nexus`.

```
RUN ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} \
${NEXUS_HOME}/nexus2
```

- Faites de `/opt/nexus` le répertoire de travail courant :

```
WORKDIR ${NEXUS_HOME}
```

**chapitre 9 |** Révision complète

- Définissez un point de montage de volume pour stocker les données persistantes du serveur Nexus :

```
VOLUME ["/opt/nexus/sonatype-work"]
```

- Définissez l'instruction CMD sur le fichier de script nexus-start.sh.

```
CMD ["sh", "nexus-start.sh"]
```

Une fois terminé, le Containerfile se présente comme suit :

```
FROM ubi8/ubi:8.5

MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus

RUN yum install -y --setopt=tsflags=nodocs java-1.8.0-openjdk-devel \
 && yum clean all -y

RUN groupadd -r nexus -f -g 1001 \
 && useradd -u 1001 -r -g nexus -m -d ${NEXUS_HOME} -s /sbin/nologin \
 -c "Nexus User" nexus \
 && chown -R nexus:nexus ${NEXUS_HOME} \
 && chmod -R 755 ${NEXUS_HOME}

USER nexus

ADD nexus-${NEXUS_VERSION}-bundle.tar.gz ${NEXUS_HOME}
ADD nexus-start.sh ${NEXUS_HOME}/

RUN ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2
WORKDIR ${NEXUS_HOME}

VOLUME ["/opt/nexus/sonatype-work"]

CMD ["sh", "nexus-start.sh"]
```

1.3. Compilez l'image du conteneur avec le nom **nexus**.

```
[student@workstation image]$ podman build --layers=false -t nexus .
STEP 1: FROM ubi8/ubi:8.5
Getting image source signatures
...output omitted...
STEP 13: COMMIT ...output omitted...localhost/nexus:latest
...output omitted...
```

2. Compilez et testez l'image de conteneur à l'aide de Podman avec un montage de volume :

**chapitre 9 |** Révision complète

- Utilisez le script `/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh` pour démarrer un nouveau conteneur avec un montage de volume.
- Consultez les journaux du conteneur pour vérifier que le serveur est démarré et en cours d'exécution.
- Testez l'accès au service de conteneur à l'aide de l'URL : `http://127.0.0.1:18081/nexus`.
- Supprimez le conteneur de test.

2.1. Exécutez le script `run-persistent.sh`. Remplacez le nom du conteneur comme indiqué dans le résultat de la commande `podman ps`.

```
[student@workstation images]$ cd /home/student/D0180/labs/comprehensive-review
[student@workstation comprehensive-review]$ cd deploy/local
[student@workstation local]$./run-persistent.sh
80970007036bbb313d8eeb7621fada0ed3f0b4115529dc50da4dccef0da34533
```

2.2. Consultez les journaux du conteneur pour vérifier que le serveur est démarré et en cours d'exécution.

```
[student@workstation local]$ podman ps \
> --format="{{.ID}} {{.Names}} {{.Image}}"
81f480f21d47 inspiring_poincare localhost/nexus:latest
[student@workstation local]$ podman logs inspiring_poincare | grep JettyServer
...output omitted...
... INFO [jetty-main-1] ...jetty.JettyServer - Running
... INFO [main] ...jetty.JettyServer - Started
```

2.3. Utilisez la commande `curl` pour tester le conteneur à l'aide de l'URL : `http://127.0.0.1:18081/nexus`.

```
[student@workstation local]$ curl -v 127.0.0.1:18081/nexus/ 2>&1 \
> | grep -E 'HTTP|<title>'
> GET /nexus/ HTTP/1.1
< HTTP/1.1 200 OK
<title>Nexus Repository Manager</title>
```

2.4. Supprimez le conteneur de test.

```
[student@workstation local]$ podman rm -f inspiring_poincare
81f480f21d475af683b4b003ca6e002d37e6aaa581393d3f2f95a1a7b7eb768b
```

3. Déployez l'image de conteneur de serveur Nexus sur le cluster OpenShift. Vous devez effectuer les actions suivantes :

- Ajoutez la balise `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest` à l'image de conteneur du serveur Nexus et envoyez-la (par push) au référentiel public sur quay.io.
- Créez un projet OpenShift avec le nom suivant :  `${RHT_OCP4_DEV_USER} - review`.

**chapitre 9 |** Révision complète

- Modifiez le fichier `deploy/openshift/resources/nexus-deployment.yaml` et remplacez `RHT_OCP4_QUAY_USER` par votre nom d'utilisateur Quay. Créez les ressources Kubernetes
- Créez une route pour le service Nexus. Vérifiez que vous pouvez accéder à `http://nexus-$\{RHT_OCP4_DEV_USER\}-review.\$\{RHT_OCP4_WILDCARD_DOMAIN\}/nexus/` à partir de `workstation`.

3.1. Connectez-vous à votre compte Quay.io.

```
[student@workstation local]$ podman login -u $\{RHT_OCP4_QUAY_USER\} quay.io
Password: your_quay_password
Login Succeeded!
```

3.2. Publiez l'image de conteneur du serveur Nexus sur votre registre `quay.io`.

```
[student@workstation local]$ podman push localhost/nexus:latest \
> quay.io/$\{RHT_OCP4_QUAY_USER\}/nexus:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

3.3. Les référentiels créés en envoyant par push des images à `quay.io` sont privés par défaut. Reportez-vous à la section `Repositories Visibility` de l'annexe C pour obtenir des informations sur la façon de modifier la visibilité des référentiels.

3.4. Créez le projet OpenShift :

```
[student@workstation local]$ cd ~/DO180/labs/comprehensive-review/deploy/openshift
[student@workstation openshift]$ oc login -u $\{RHT_OCP4_DEV_USER\} \
> -p $\{RHT_OCP4_DEV_PASSWORD\} $\{RHT_OCP4_MASTER_API\}
Login successful.
...output omitted...
[student@workstation openshift]$ oc new-project $\{RHT_OCP4_DEV_USER\}-review
Now using project ...output omitted...
```

3.5. Remplacez `RHT_OCP4_QUAY_USER` dans le fichier de ressources par votre nom d'utilisateur Quay et créez les ressources Kubernetes :

```
[student@workstation openshift]$ export RHT_OCP4_QUAY_USER
[student@workstation openshift]$ envsubst < resources/nexus-deployment.yaml \
> | oc create -f -
service/nexus created
persistentvolumeclaim/nexus created
deployment.apps/nexus created
[student@workstation openshift]$ oc get pods
NAME READY STATUS RESTARTS AGE
nexus-77c479bb4f-kscz7 0/1 ContainerCreating 0 11s
```

3.6. Exposez le service en créant une route :

```
[student@workstation openshift]$ oc expose svc/nexus
route.route.openshift.io/nexus exposed.
[student@workstation openshift]$ oc get route -o yaml
apiVersion: v1
items:
- apiVersion: route.openshift.io/v1
 kind: Route
 ...output omitted...
 spec:
 host: nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}
...output omitted...
```

- 3.7. Utilisez un navigateur pour vous connecter à l’application Web du serveur Nexus à l’adresse `http://nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/`.

## Évaluation

Après avoir déployé l’image de conteneur du serveur Nexus sur le cluster OpenShift, vérifiez votre travail en exécutant le script de notation de l’atelier :

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

## Fin

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

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

L’atelier est maintenant terminé.



## annexe A

# Implémentation de l'architecture des microservices

**Objectif** Refactoriser une application en microservices.

**Résultats**

- Diviser une application entre plusieurs conteneurs pour séparer les couches et les services distincts.

**Sections**

- Implémentation de l'architecture des microservices (et exercice guidé)

# Implémentation des architectures des microservices

## Résultats

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

- Diviser une application entre plusieurs conteneurs pour séparer les couches et les services distincts.
- Décrire des approches typiques pour décomposer une application monolithique en plusieurs unités déployables.
- Décrire comment diviser l'application To Do List en trois conteneurs correspondant à ses niveaux logiques.

## Avantages de la décomposition d'une application monolithique en conteneurs

Le développement d'application traditionnel présente généralement de nombreuses fonctions distinctes regroupées en une seule unité de déploiement ou en une application monolithique. Le développement traditionnel peut également déployer des services de support, tels que des bases de données et d'autres services de middleware, sur le même serveur que l'application. Bien que les applications monolithiques puissent toujours être déployées dans un conteneur, nombreux sont les avantages d'une architecture de conteneur, tels que l'évolutivité et l'agilité, ne sont pas aussi répandus. La dissociation des monolithes requiert une attention particulière et il est recommandé que dans les applications de microservices que chaque microservice exécute la fonctionnalité minimale pouvant être exécutée séparément sur chaque conteneur.

Avoir des conteneurs plus petits et décomposer une application et ses services de support en plusieurs conteneurs offre de nombreux avantages, tels que :

- Une meilleure utilisation du matériel, car les petits conteneurs sont plus faciles à intégrer dans la capacité de l'hôte disponible.
- Une mise à l'échelle plus facile, car certaines parties de l'application peuvent être mises à l'échelle pour gérer une charge de travail accrue sans avoir à mettre à l'échelle d'autres parties de l'application.
- Des mises à niveau plus faciles, car les développeurs peuvent mettre à jour des parties de l'application sans affecter d'autres parties de la même application.

Voici deux façons populaires de scinder une application :

- Niveaux : sur la base de couches architecturales.
- Services : sur la base des fonctionnalités d'application.

## Division en fonction des couches (niveaux)

Il arrive souvent que les développeurs organisent les applications en niveaux, selon la proximité des fonctions avec les utilisateurs et la distance par rapport aux magasins de données. Voici un bon exemple de l'architecture traditionnelle à trois niveaux : présentation, logique métier et persistance.

**annexe A |** Implémentation de l'architecture des microservices

Cette architecture logique correspond généralement à une architecture de déploiement physique, dans laquelle la couche de présentation serait déployée sur un serveur Web, la couche métier sur un serveur d'applications et la couche de persistance sur un serveur de base de données.

La décomposition d'une application en niveaux permet aux développeurs de se spécialiser en technologies particulières basées sur les niveaux de l'application. Par exemple, certains développeurs se concentrent sur les applications Web, tandis que d'autres préfèrent le développement de bases de données. Un autre avantage est la possibilité de fournir des implémentations de niveau alternatif basées sur différentes technologies ; par exemple, la création d'une application mobile comme autre service frontal pour une application existante. L'application mobile serait un niveau de présentation alternatif, réutilisant les niveaux métier et de persistance de l'application Web d'origine.

Pour les plus petites applications, les niveaux de présentation et métier sont généralement déployés comme une simple unité. Par exemple sur le même serveur Web, mais lorsque la charge augmente, la couche de présentation est déplacée vers sa propre unité de déploiement pour répartir la charge. Des applications plus petites peuvent même intégrer la base de données. Les développeurs compilent et déploient souvent des applications plus exigeantes de cette façon monolithique.

Lorsque les développeurs divisent une application monolithique en niveaux, ils appliquent généralement plusieurs modifications :

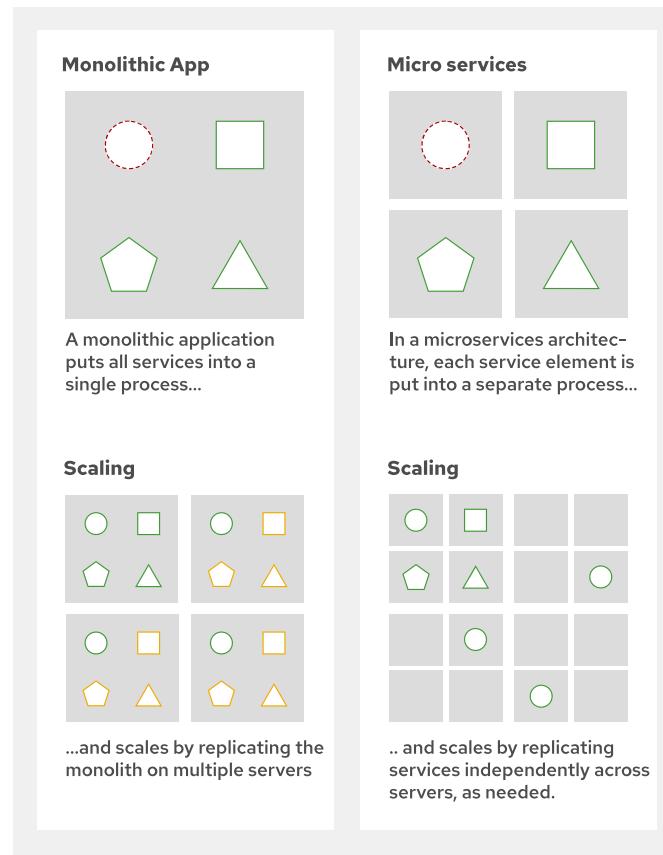
- Les paramètres de connexion à une base de données et à d'autres services de middleware, tels que la messagerie, ont été codés en dur sur des adresses IP fixes ou des noms d'hôte, généralement `localhost`. Ils doivent être paramétrés pour pointer vers des serveurs externes qui peuvent différer du développement à la production.
- Dans le cas des applications Web, les appels Ajax ne peuvent pas être effectués à l'aide d'URL relatives. Ils doivent utiliser une URL absolue pointant vers un nom d'hôte DNS public fixe.
- Les navigateurs Web modernes refusent les appels Ajax vers des serveurs différents de celui contenant le script qui effectue l'appel, par mesure de sécurité. L'application doit avoir des permissions pour le partage des ressources de différentes origines (CORS).

Une fois que les niveaux d'application sont divisés pour pouvoir s'exécuter à partir de différents serveurs, il ne devrait pas y avoir de problème pour les exécuter à partir de différents conteneurs.

## Division basée sur des services discrets

La plupart des applications complexes sont composées de nombreux services semi-indépendants. Par exemple, un magasin en ligne aurait un catalogue de produits, un panier, un système de paiement, un système d'expédition, etc.

Lorsqu'un service donné dans une application monolithique se dégrade, la mise à l'échelle du service pour améliorer les performances implique la mise à l'échelle de tous les autres services d'application constitutifs. Si toutefois le service dégradé fait partie d'une architecture de microservices, le service affecté est mis à l'échelle indépendamment des autres services d'application. La figure suivante illustre la mise à l'échelle des services à la fois pour une architecture monolithique et basée sur des microservices :



**Figure A.1: Comparaison de la mise à l'échelle d'applications dans une architecture monolithique par rapport à une architecture de microservices**



### Note

Dans le schéma précédent :

- Les figures en pointillés rouges représentent les services qui sont surutilisés.
- Les figures en jaune représentent les services qui sont sous-utilisés.
- Les figures en vert représentent les services qui sont correctement dimensionnés.

Les architectures orientées services (SOA) traditionnelles et les architectures de microservices plus récentes compilent et déplacent ces ensembles de fonctions en tant qu'unités distinctes. Cela permet à chaque ensemble de fonctions d'être développé par sa propre équipe, d'être mis à jour et d'évoluer sans perturber les autres ensembles de fonctions (ou services). Des problèmes interfonctionnels tels que l'authentification peuvent également être empaquetés et déployés en tant que services utilisés par d'autres implémentations de service.

La division de chaque problème en un serveur séparé peut entraîner la création de nombreuses applications. Elles sont conçues, empaquetées et déployées de façon logique en un petit nombre d'unités, parfois même sous la forme d'une seule unité monolithique utilisant une approche de service.

**annexe A |** Implémentation de l'architecture des microservices

Les conteneurs permettent de matérialiser des architectures basées sur des services lors du déploiement. C'est la raison pour laquelle les microservices et les conteneurs vont généralement de pair. Toutefois, les conteneurs seuls ne suffisent pas ; ils doivent être complétés par des outils d'orchestration pour gérer les dépendances entre les services.

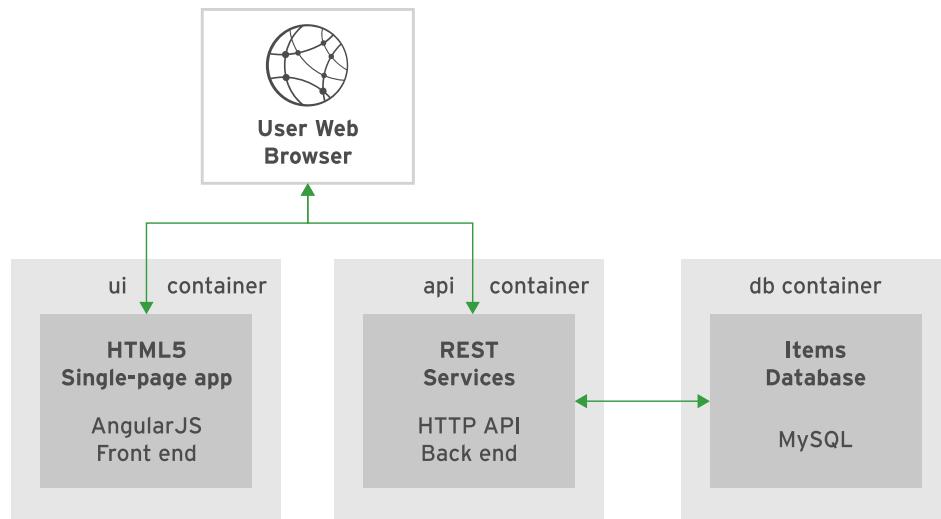
L'architecture de microservices porte les architectures basées sur les services à leur extrême. Un service est aussi petit que possible (sans scinder un ensemble de fonctions) et déployé et géré comme une unité indépendante, plutôt que comme élément d'une plus grande application. Cela permet de réutiliser les microservices existants pour créer de nouvelles applications.

Pour scinder une application en services, il faut le même type de changement que lors de la scission en niveaux ; par exemple, configurez les paramètres de connexion vers des bases de données et d'autres services de middleware, et gérez les protections de sécurité des navigateurs Web.

## Refactorisation de l'application To Do List

L'application To Do List est une application simple avec un seul jeu de fonctions. Par conséquent, sa fragmentation en services n'a pas vraiment de sens. Toutefois, sa refactorisation en niveaux de présentation et métier, c'est-à-dire en un service frontal et un service backend à déployer dans des conteneurs distincts, illustre le type de changements que la division d'une application typique en services aurait besoin.

La figure suivante montre l'application To Do List déployée en trois conteneurs, un pour chaque niveau :



**Figure A.2: Application To Do List scindée en plusieurs niveaux et chacun déployé en tant que conteneur**

En comparant le code source de l'application monolithique d'origine à celle refactorisée, voici un aperçu des modifications :

- The front-end JavaScript in `script/items.js` uses `workstation.lab.example.com` as the host name to reach the back end.
- Le backend utilise des variables d'environnement pour obtenir les paramètres de connexion à la base de données.
- The back end has to reply to requests using the HTTP `OPTIONS` verb with headers telling the web browser to accept requests coming from different DNS domains using CORS .

D'autres versions du service principal peuvent avoir des modifications similaires. Chaque langage de programmation et chaque structure REST a sa propre syntaxe et ses propres fonctions.



## Références

### **Page d'application monolithique dans Wikipedia**

[https://en.wikipedia.org/wiki/Monolithic\\_application](https://en.wikipedia.org/wiki/Monolithic_application)

### **Page CORS dans Wikipédia**

[https://en.wikipedia.org/wiki/Cross-origin\\_resource\\_sharing](https://en.wikipedia.org/wiki/Cross-origin_resource_sharing)

## ► Exercice guidé

# Refactorisation de l'application To Do List

Au cours de cet atelier, vous allez refactoriser l'application To Do List en plusieurs conteneurs reliés entre eux, permettant à l'application HTML 5 frontale, à l'API REST Node.js et au serveur MySQL de s'exécuter dans leurs propres conteneurs.

## Résultats

Vous devez pouvoir refactoriser une application monolithique dans ses niveaux et déployer chaque niveau en tant que microservice.

## Avant De Commencer

Exécutez la commande suivante pour configurer les répertoires de travail pour l'atelier avec les fichiers de l'application To Do List :

```
[student@workstation ~]$ lab appendix-microservices start
```

## Instructions

### ► 1. Déplacer les fichiers HTML

La première étape de la refactorisation de l'application To Do List consiste à déplacer le code frontal depuis l'application vers son propre conteneur en cours d'exécution. Cette étape vous guide lors du déplacement de l'application HTML et de ses fichiers dépendants dans leur propre répertoire en vue de leur déploiement sur un serveur Apache exécuté dans un conteneur.

- 1.1. Déplacez les fichiers HTML et statiques vers le répertoire `src/` à partir de l'application To Do List Node.js monolithique :

```
[student@workstation ~]$ cd ~/D0180/labs/appendix-microservices/apps/html5/
[student@workstation html5]$ mv \
> ~/D0180/labs/appendix-microservices/apps/nodejs/todo/* \
> ~/D0180/labs/appendix-microservices/apps/html5/src/
```

- 1.2. L'application frontale actuelle interagit avec le service API à l'aide d'une URL relative. Étant donné que l'API et le code frontal vont désormais s'exécuter dans des conteneurs distincts, le code frontal doit être adapté pour pointer vers l'URL absolue de l'API de l'application To Do List.

Ouvrez le fichier `/home/student/D0180/labs/appendix-microservices/apps/html5/src/script/item.js`. Au bas du fichier, recherchez la méthode suivante :

```
app.factory('itemService', function ($resource) {
 return $resource('api/items/:id');
});
```

Remplacez ce code par le contenu suivant :

```
app.factory('itemService', function ($resource) {
 return $resource('http://workstation.lab.example.com:30080/todo/api/
items/:id');
});
```

Assurez-vous qu'il n'y a pas de saut de ligne dans la nouvelle URL, enregistrez le fichier et quittez l'éditeur.

## ► 2. Créer l'image HTML

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

```
[student@workstation html5]$ podman login registry.redhat.io
Username: your_username
Password: your_password
Login Succeeded!
```

### 2.2. Créez l'image Apache enfant :

```
[student@workstation html5]$ cd ~/D0180/labs/appendix-microservices/deploy/html5
[student@workstation html5]$./build.sh
STEP 1: FROM registry.redhat.io/rhel8/httpd-24:1
Getting image source signatures
Copying blob 1b6a9fa02570 done
...output omitted...
Storing signatures
STEP 2: COPY ./src/ ${HOME}/
STEP 3: COMMIT do180/todo_frontend
dda10a4bf12ff987331e482e5cd87658abc5724da4b5b7d136874a9e471acf71
```

### 2.3. Vérifiez que l'image a été générée correctement :

```
[student@workstation html5]$ podman images
REPOSITORY TAG IMAGE ID CREATED
SIZE
localhost/do180/todo_frontend latest dda10a4bf12f About a minute ago
438 MB
registry.redhat.io/rhel8/httpd-24 1 adcf72f31a0b 13 days ago
438 MB
...
```

## ► 3.Modifier l'API REST pour se connecter à des conteneurs externes

### 3.1. L'API REST utilise actuellement des valeurs codées en dur pour se connecter à la base de données MySQL. Modifiez le fichier /home/student/D0180/labs/appendix-microservices/apps/nodejs/models/db.js, qui contient la configuration de la base de données. Mettre à jour le `dbname`, `username`, et `password` valeurs pour utiliser des variables d'environnement à la place. En outre, mettez à jour `params.host` pour qu'il pointe vers le nom de l'hôte exécutant le conteneur MySQL et mettez à jour `params.port` pour refléter le port redirigé vers le conteneur. Les deux valeurs sont disponibles en tant que variables d'environnement

**annexe A |** Implémentation de l'architecture des microservices

MySQL\_SERVICE\_HOST et MySQL\_SERVICE\_PORT, respectivement. Le contenu remplacé doit ressembler à ceci :

```
module.exports.params = {
 dbname: process.env.MYSQL_DATABASE,
 username: process.env.MYSQL_USER,
 password: process.env.MYSQL_PASSWORD,
 params: {
 host: process.env.MYSQL_SERVICE_HOST,
 port: process.env.MYSQL_SERVICE_PORT,
 dialect: 'mysql'
 }
};
```

**Note**

Ce fichier peut être copié et collé à partir de /home/student/D0180/solutions/appendix-microservices/apps/nodejs/models/db.js.

- 3.2. Configurez le backend pour gérer le partage des ressources de différentes origines (CORS). Cela se produit lorsqu'une demande de ressource est faite à partir d'un domaine différent de celui de la demande. Étant donné que l'API doit gérer les demandes provenant d'un domaine DNS différent (l'application frontale), il est nécessaire de créer des exceptions de sécurité pour permettre à ces demandes de réussir. Apportez les modifications suivantes à l'application dans l'éditeur de votre choix afin de gérer le partage CORS.

Ajoutez "restify-cors-middleware": "1.1.1" en tant que nouvelle dépendance du fichier package.json figurant dans /home/student/D0180/labs/appendix-microservices/apps/nodejs/package.json. N'oubliez pas de placer une virgule à la fin de la dépendance précédente. Assurez-vous que la fin du fichier se présente comme suit :

```
"sequelize": "5.21.1",
"mysql2": "2.0.0",
"restify-cors-middleware": "1.1.1"
}
```

Mettez à jour le fichier app.js situé à l'emplacement /home/student/D0180/labs/appendix-microservices/apps/nodejs/app.js pour configurer l'utilisation de CORS. Exigez le module restify-cors-middleware sur la deuxième ligne, puis mettez à jour le contenu du fichier pour qu'il corresponde à ce qui suit :

```
var restify = require('restify');
var corsMiddleware = require('restify-cors-middleware');
var controller = require('./controllers/items');
...output omitted...
var server = restify.createServer()
 .use(restify.plugins.fullResponse())
 .use(restify.plugins.queryParser())
 .use(restify.plugins.bodyParser());
```

**annexe A |** Implémentation de l'architecture des microservices

```
const cors = corsMiddleware({ origins: ['*'] });

server.pre(cors.preflight);
server.use(cors.actual);

controller.context(server, '/todo/api', model);
```

Les origines avec la valeur `["*"]` indiquent au serveur d'autoriser tous les domaines. Dans un serveur de production, cette valeur est généralement une matrice de domaines connus pour exiger l'accès à l'API.

**▶ 4. Compiler l'image REST API**

- 4.1. Générez l'image enfant de l'API REST à l'aide de la commande suivante. Cette image utilise l'image Node.js.

```
[student@workstation html5]$ cd ~/DO180/labs/appendix-microservices/deploy/nodejs
[student@workstation nodejs]$./build.sh
STEP 1: FROM quay.io/redhattraining/do180-todonodejs-12
Getting image source signatures
...output omitted...
STEP 6: CMD ["/bin/bash", "-c", "./run.sh"]
STEP 7: COMMIT do180/todonodejs
18f8d5dd8e25ed19a54750c8b96ae075adf437f5f0ba5ebbf7b9fe4b75526b3
```

- 4.2. Exécutez la commande `podman images` pour vérifier que toutes les images requises ont été compilées correctement :

```
[student@workstation nodejs]$ podman images
REPOSITORY TAG IMAGE ID CREATED
SIZE
localhost/do180/todonodejs latest 18f8d5dd8e25 About a
minute ago 852 MB
localhost/do180/todo_frontend latest dda10a4bf12f 10 minutes
ago 438 MB
...output omitted...
```

**▶ 5. Exécutez les conteneurs**

- 5.1. Utilisez le script `run.sh` pour exécuter les conteneurs :

```
[student@workstation nodejs]$ cd linked/
[student@workstation linked]$./run.sh
• Creating database volume: OK
• Launching database: OK
• Importing database: OK
• Launching To Do application: OK
```

- 5.2. Exécutez la commande `podman ps` pour confirmer que les trois conteneurs sont en cours d'exécution :

```
[student@workstation linked]$ podman ps
... IMAGE ... PORTS NAMES
... localhost/do180/todo_frontend:latest ... 0.0.0.0:30000->80/tcp todo_frontend
... localhost/do180/todonodejs:latest ... 0.0.0.0:30080->30080/tcp todoapi
... registry.redhat.io/rhel8/mysql-80:1 ... 0.0.0.0:30306->3306/tcp mysql
```

## ▶ 6. Tester l'application

- 6.1. Utilisez la commande `curl` pour vérifier que l'API REST de l'application To Do List fonctionne correctement :

```
[student@workstation linked]$ curl -w "\n" 127.0.0.1:30080/todo/api/items/1
{"description": "Pick up newspaper", "done": false, "id":1}
```

- 6.2. Ouvrez Firefox sur la machine `workstation` et accédez à 127.0.0.1:30000, où vous devriez voir l'application To Do List.
- 6.3. Revenez au dossier personnel de l'utilisateur.

```
[student@workstation linked]$ cd ~
[student@workstation ~]$
```

## Fin

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

```
[student@workstation ~]$ lab appendix-microservices finish
```

L'exercice guidé est maintenant terminé.

# Résumé

---

Dans ce chapitre, vous avez appris les principes suivants :

- La scission d'une application monolithique en plusieurs conteneurs permet une meilleure évolutivité des applications, facilite les mises à niveau et permet une utilisation plus importante du matériel.
- Les trois niveaux communs pour la division logique d'une application sont le niveau de présentation, le niveau métier et le niveau de persistance.
- Le partage de ressources de différentes origines (Cross-Origin Resource Sharing – CORS) peut empêcher les appels Ajax vers des serveurs différents de ceux d'où les pages ont été téléchargées. Veillez à prendre des dispositions pour autoriser CORS à partir d'autres conteneurs dans l'application.
- Les images de conteneur sont censés être immuables, mais les configurations peuvent être transmises soit au moment de la compilation de l'image, soit en créant un stockage persistant pour les configurations.

## annexe B

# 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 *publics* 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 *publics* 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**.

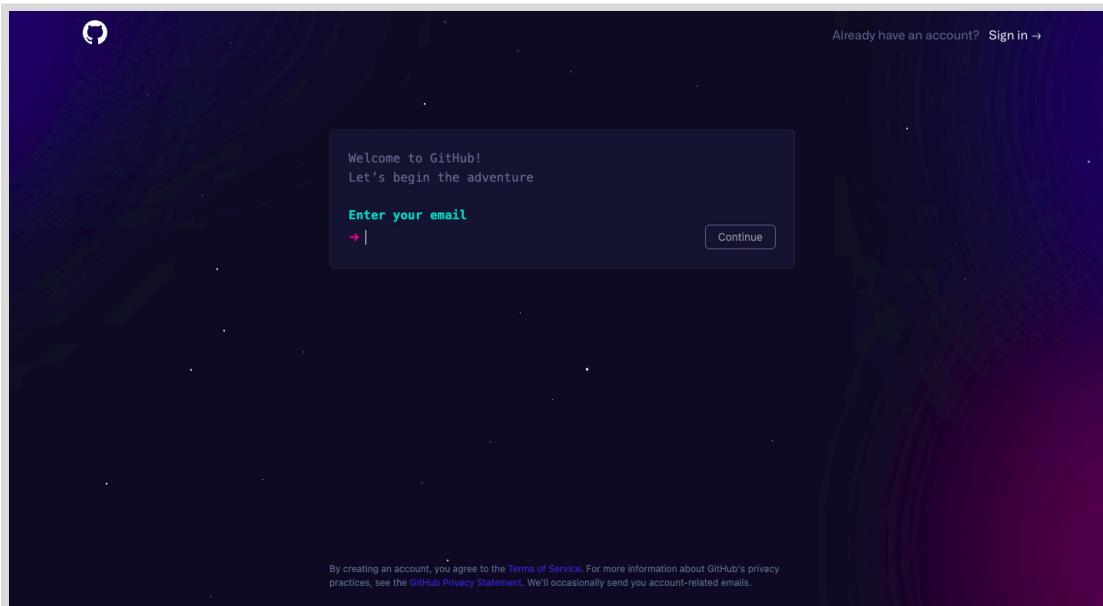
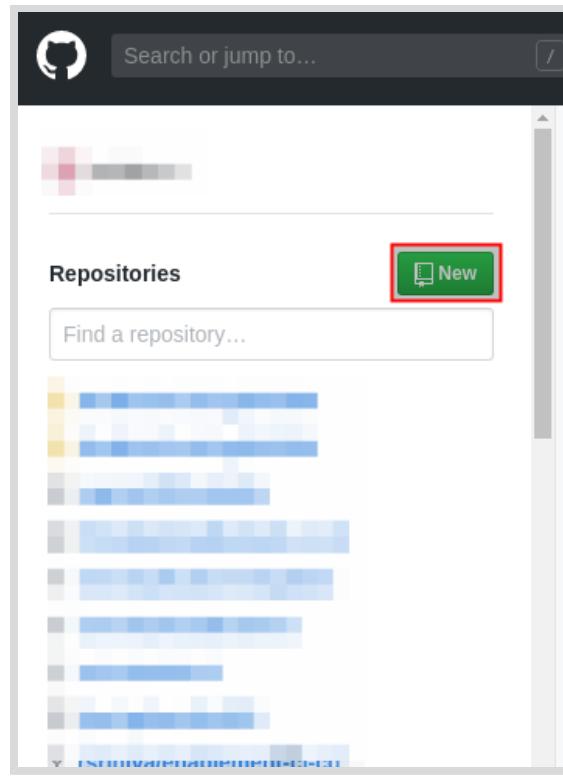


Figure B.1: Création d'un compte GitHub

3. Vous recevrez 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.

**annexe B |** Création d'un compte GitHub

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 B.2: Crédit d'un nouveau référentiel Git**

Sinon, cliquez sur l'icône plus (+) dans le coin supérieur droit (à droite de l'icône de cloche), puis cliquez sur **New repository**.

**annexe B** | Création d'un compte GitHub

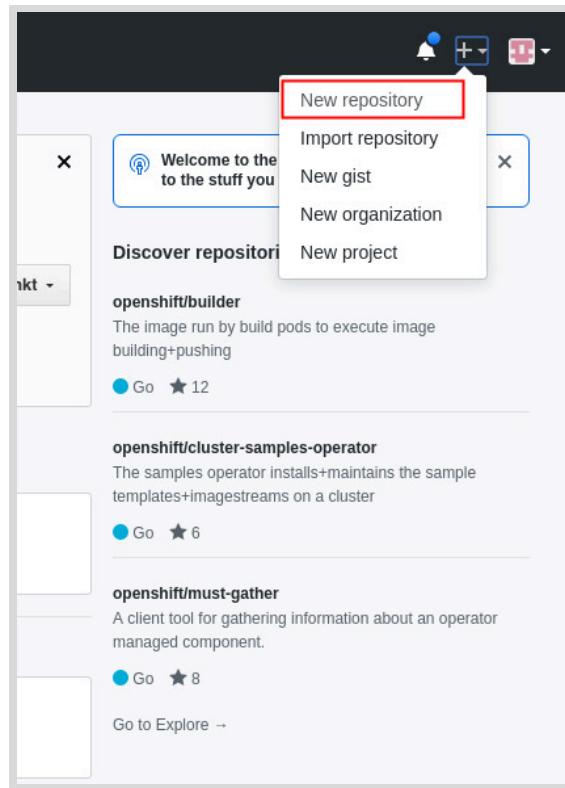


Figure B.3: Crédation d'un nouveau référentiel Git



## Références

### **Signing up for a new GitHub account**

<https://help.github.com/en/articles/signing-up-for-a-new-github-account>

## annexe C

# 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 D).

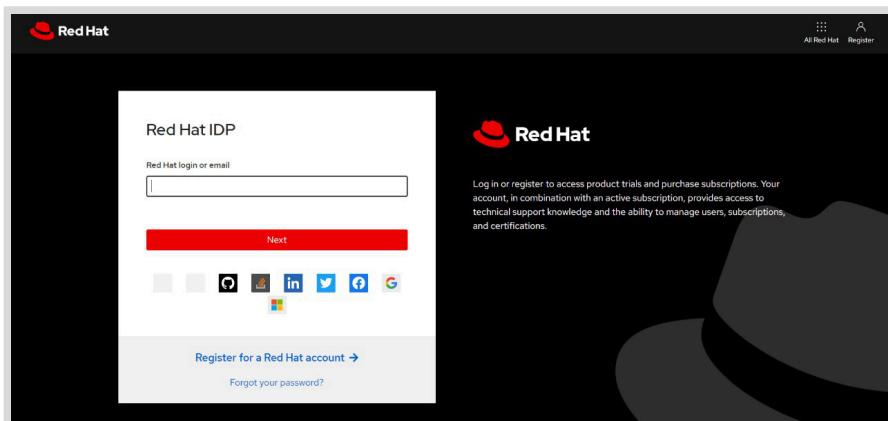


Figure C.1: Connectez-vous à l'aide de vos informations d'identification Red Hat.

Après vous être connecté à Quay, vous pouvez créer de nouveaux référentiels d'images en cliquant sur **Create New Repository** dans la page **Repositories**.

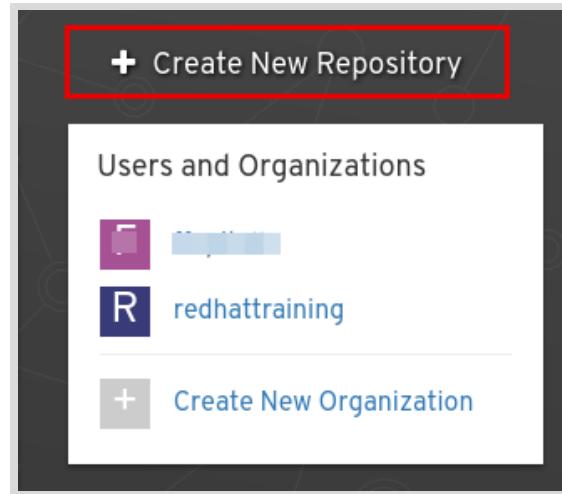


Figure C.2: Création d'un référentiel d'images

Si non, cliquez sur l'icône plus (+) dans le coin supérieur droit (à gauche de l'icône de cloche), puis cliquez sur **New Repository**.

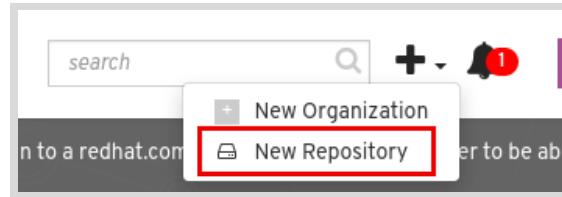


Figure C.3: Création d'un référentiel d'images

## 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 le lien *Change 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.

The screenshot shows the Quay.io interface at the URL <https://quay.io/repository/>. The top navigation bar includes links for RED HAT® Quay.io, EXPLORE, APPLICATIONS, REPOSITORIES (which is highlighted in red), and TUTORIAL. Below the navigation is a dark header with the word "Repositories". On the left, there's a "Starred" section with a star icon and the text "Starred". A message says "You haven't starred any repositories yet." with a subtext "Stars allow you to easily access your favorite repositories.". Below this, there's a section titled "RHT\_OCP4\_QUAY\_USER" with a small profile picture. Four repository cards are listed:

- do180-mysql-57-rhel7
- do180-todonodejs
- do180-quote-php
- nexus

Each card has a star icon to its right. At the bottom right of the page, there are two grid icons: one with 4 squares and one with 9 squares.

## 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**.

**annexe C |** Création d'un compte Quay

The screenshot shows the Quay.io repository settings interface. At the top, there is a message: "Trust Disabled" with the subtext "Signing is disabled on this repository." and a "Enable Trust" button. Below this is the "Events and Notifications" section, which states "No notifications have been setup for this repository." and "Click the 'Create Notification' button above to add a new notification for a repository event." The "Repository Visibility" section indicates the repository is "private" and has a "Make Public" button. The "Delete Repository" section contains a warning: "Deleting a Repository cannot be undone. Here be dragons!" and a red "Delete Repository" button.

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 D

# 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

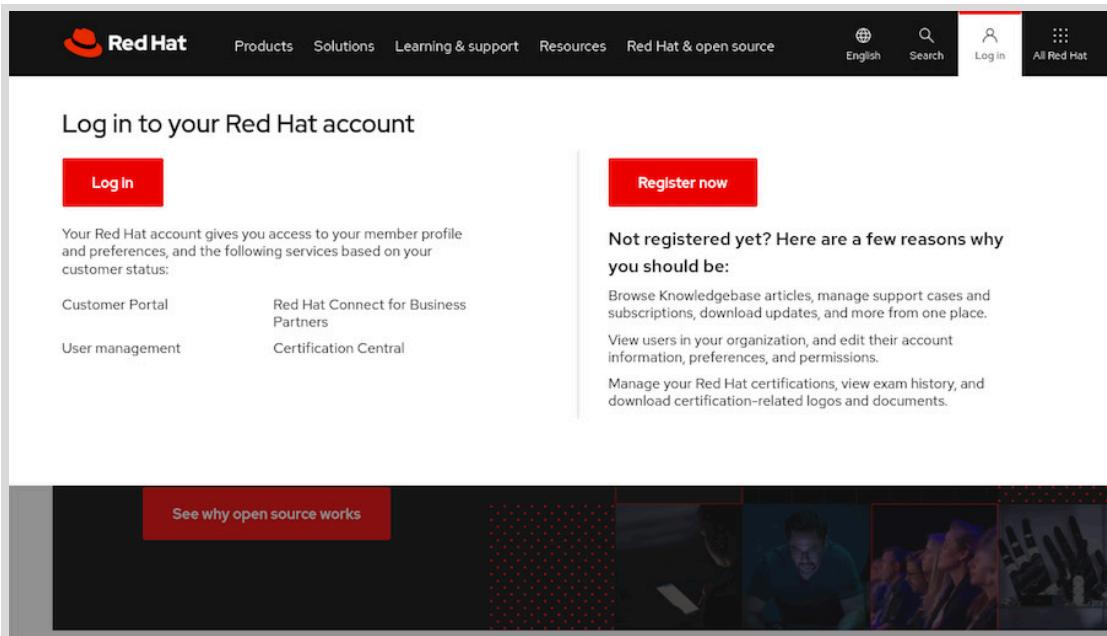
À la fin de cette section, vous devez pouvoir créer un compte Red Hat pour accéder aux images Red Hat Container Catalog pour les ateliers de ce cours.

### 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 :

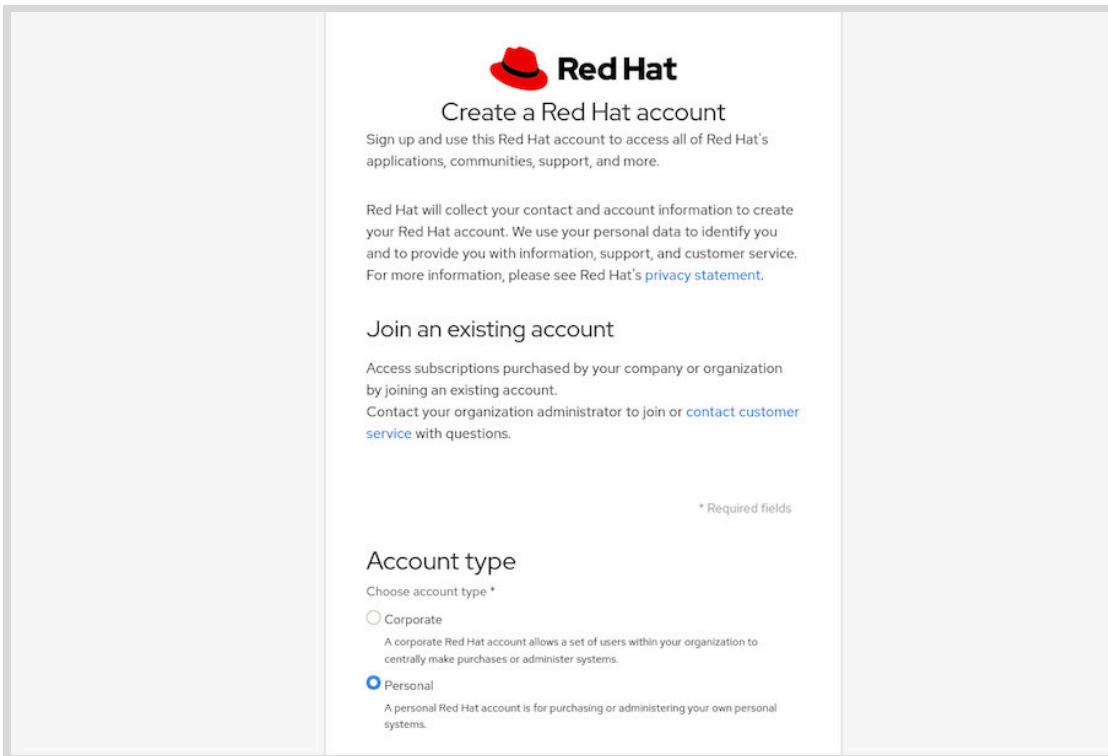
1. Accédez à <https://redhat.com> à l'aide d'un navigateur Web.
2. Cliquez sur **Log in** en haut à droite.
3. Cliquez sur **Register now** dans le volet de droite.



**Figure D.1: Enregistrement d'un nouveau compte**

4. Définissez Account type sur Personal.

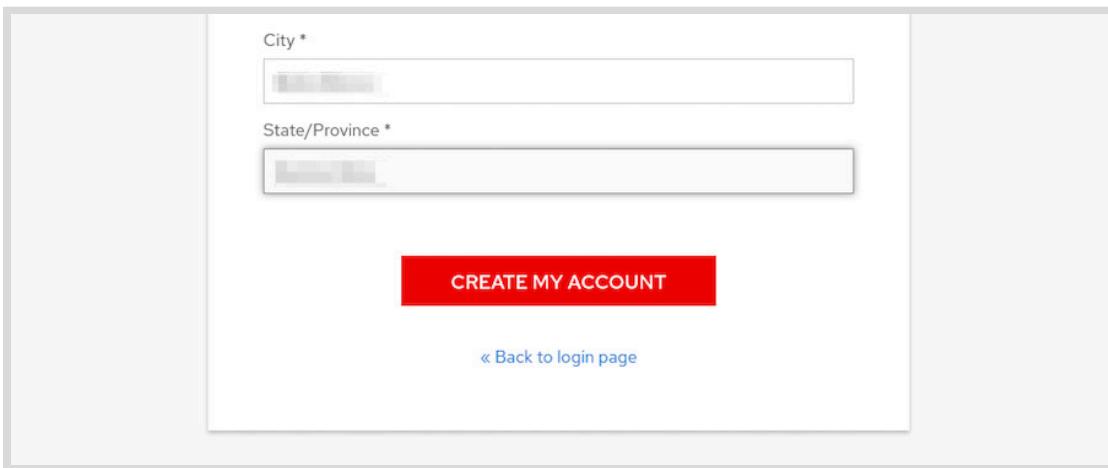
**annexe D |** Création d'un compte Red Hat



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 subtext explains: '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' provides instructions for accessing company subscriptions and contacting customer service. A note at the bottom right indicates '\* Required fields'. The 'Account type' section follows, with a subtext 'Choose account type \*'. It contains two radio button options: 'Corporate' (unselected) and 'Personal' (selected). A detailed description for 'Personal' accounts is provided: 'A personal Red Hat account is for purchasing or administering your own personal systems.'

**Figure D.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 form for entering personal information. It includes fields for 'City \*' (with a blurred input field), 'State/Province \*' (with a blurred input field), and a large red 'CREATE MY ACCOUNT' button. Below the button is a link '« Back to login page'.

**Figure D.3: Renseigner le formulaire d'informations personnelles**

==



## Références

### Portail client Red Hat

<https://access.redhat.com/>



## annexe E

# 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 parcourrez 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 :

1. 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
```

2. Change to your local clone of the D0180-apps repository and switch to the `master` branch:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
```

3. Supprimez votre branche locale :

```
[student@workstation D0180-apps]$ git branch -d your-branch
```

4. Supprimez la branche distante sur votre compte GitHub personnel :

```
[student@workstation D0180-apps]$ git push origin --delete your-branch
```

**annexe E |** Commandes Git utiles

5. Utilisez la sous-commande **start** pour redémarrer l'exercice :

```
[student@workstation D0180-apps]$ cd ~
[student@workstation ~]$ lab your-exercise start
```

## 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
```

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
```

3. Basculez vers la branche **master** de votre référentiel local :

```
[student@workstation D0180-apps]$ git checkout master
```

4. Supprimez votre branche locale :

```
[student@workstation D0180-apps]$ git branch -d your-branch
```

5. Supprimez la branche distante sur votre compte GitHub personnel :

```
[student@workstation D0180-apps]$ git push origin --delete your-branch
```

6. Vous pouvez maintenant redémarrer l'exercice en exécutant sa sous-commande **start** :

```
[student@workstation D0180-apps]$ cd ~
[student@workstation ~]$ lab your-exercise start
```

## 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 utilisiez la totalité de votre quota alloué sur le fournisseur de cloud et sur le cluster OpenShift que vous partagez avec d'autres stagiaires. 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.

**annexe E |** Commandes Git utiles

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 ~/DO180-apps
[student@workstation DO180-apps]$ git commit -a -m "Paused at step X.Y"
[student@workstation DO180-apps]$ git push
```

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
```

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
```

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>