



Red Hat OpenShift I: Containers & Kubernetes



OCP 4.5 DO180
Red Hat OpenShift I: Containers & Kubernetes
Edición 2 20200911
fecha de publicación 20200911

Autores: Zach Guterman, Dan Kolepp, Eduardo Ramirez Ronco,
Jordi Sola Alaball, Richard Allred
Editor: Seth Kenlon, Dave Sacco, Connie Petlitzer

Copyright © 2019 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are
Copyright © 2019 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, Hibernate, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered 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 and/or its affiliates.

XFS® is a registered trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

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.

Colaboradores: Michael Jarrett, Forrest Taylor y Manuel Aude Morales

Convenciones del documento	vii
Introducción	ix
DO180: Red Hat OpenShift I: Containers & Kubernetes	ix
Orientación sobre el entorno del aula	xi
Internacionalización	xv
1. Introducción a la tecnología de contenedores	1
Descripción general de la tecnología de contenedores	2
Cuestionario: Descripción general de la tecnología de contenedores	5
Descripción general de la arquitectura de contenedores	9
Cuestionario: Descripción general de la arquitectura de contenedores	12
Resumen de Kubernetes y OpenShift	14
Cuestionario: Descripción de Kubernetes y OpenShift	17
Ejercicio Guiado: Configuración del entorno del aula	19
Resumen	25
2. Creación de servicios contenerizados	27
Aprovisionamiento de servicios contenerizados	28
Ejercicio Guiado: Creación de una instancia de base de datos MySQL	34
Trabajo de laboratorio: Creación de servicios en contenedores	37
Resumen	42
3. Administración de contenedores	43
Administración del ciclo de vida de los contenedores	44
Ejercicio Guiado: Administración de un contenedor MySQL	52
Conexión de almacenamiento persistente a los contenedores	57
Ejercicio Guiado: Persistencia de una base de datos MySQL	60
Acceso a los contenedores	63
Ejercicio Guiado: Cargar la base de datos	67
Trabajo de laboratorio: Administración de contenedores	71
Resumen	80
4. Administración de imágenes de contenedores	81
Acceso a los registros	82
Cuestionario: Trabajo con registros	88
Manipulación de imágenes de contenedores	92
Ejercicio Guiado: Creación de una imagen de contenedor de Apache personalizada	98
Trabajo de laboratorio: Administración de imágenes	103
Resumen	111
5. Creación de imágenes de contenedores personalizadas	113
Diseño de imágenes de contenedores personalizadas	114
Cuestionario: Enfoques para el diseño de imágenes de contenedores	118
Compilación de imágenes de contenedores personalizadas con Dockerfiles	120
Ejercicio Guiado: Creación de una imagen de contenedor de Apache básica	126
Trabajo de laboratorio: Creación de imágenes de contenedores personalizadas	130
Resumen	137
6. Implementación de aplicaciones contenedorizadas en OpenShift	139
Descripción de la arquitectura de Kubernetes y OpenShift	140
Cuestionario: Descripción de Kubernetes y OpenShift	146
Creación de recursos de Kubernetes	150
Ejercicio Guiado: Implementación de un servidor de bases de datos en OpenShift	161
Creación de rutas	166
Ejercicio Guiado: Exposición de un servicio como una ruta	170
Creación de aplicaciones con Fuente a imagen (Source-to-Image)	175
Ejercicio Guiado: Creación de una aplicación en contenedores con fuente a imagen	185

Creación de aplicaciones con la consola web de OpenShift	192
Ejercicio Guiado: Creación de una aplicación con la consola web	198
Trabajo de laboratorio: Implementación de aplicaciones contenedorizadas en OpenShift	216
Resumen	221
7. Implementación de aplicaciones con múltiples contenedores	223
Consideraciones para aplicaciones con varios contenedores	224
Ejercicio Guiado: Implementación de la aplicación web y los contenedores MySQL	229
Implementación de una aplicación con varios contenedores en OpenShift	236
Ejercicio Guiado: Creación de una aplicación con una plantilla	247
Trabajo de laboratorio: Implementación de aplicaciones con múltiples contenedores	253
Resumen	260
8. Solución de problemas de aplicaciones en contenedores	261
Solución de problemas de implementaciones y compilaciones con S2I	262
Ejercicio Guiado: Solución de problemas de compilación de OpenShift	268
Solución de problemas de aplicaciones contenerizadas	276
Ejercicio Guiado: Configuración de los registros del contenedor de Apache para depuración	283
Trabajo de laboratorio: Solución de problemas de aplicaciones en contenedores	286
Resumen	297
9. Revisión exhaustiva	299
Revisión completa	300
Trabajo de laboratorio: Organizar en contenedores e implementar una aplicación de software	303
A. Implementación de la arquitectura de microservicios	313
Implementación de las arquitecturas de microservicios	314
Ejercicio Guiado: Refactorización de la aplicación To Do List (Lista de tareas)	319
Resumen	324
B. Creación de una cuenta GitHub	325
Creación de una cuenta GitHub	326
C. Creación de una cuenta Quay	329
Creación de una cuenta Quay	330
Visibilidad de repositorios	333
D. Comandos Git útiles	337
Comandos Git	338

Convenciones del documento



Referencias

En "Referencias", se describe el lugar donde se puede encontrar documentación externa relevante para un tema.



nota

Las "Notas" son consejos, atajos o enfoques alternativos para una tarea determinada. Ignorar una nota no debería tener consecuencias negativas, pero podría pasarse por alto algún truco que puede simplificar una tarea.



Importante

En las cajas (boxes) "Importante", se detallan cosas que se olvidan con facilidad: cambios de configuración que solo se aplican a la sesión actual o servicios que se deben reiniciar para poder aplicar una actualización. Ignorar una caja (box) con la etiqueta "Importante" no provocará pérdida de datos, pero puede causar irritación y frustración.



Advertencia

No se deben ignorar las "Advertencias". Es muy probable que omitir las advertencias provoque pérdida de datos.

Introducción

DO180: Red Hat OpenShift I: Containers & Kubernetes

DO180: Red Hat OpenShift I: Containers & Kubernetes es un curso práctico que enseña a los estudiantes a crear, implementar y administrar contenedores con Podman, Kubernetes y Red Hat OpenShift Container Platform.

Uno de los tenants (inquilinos) clave del movimiento de DevOps es la integración y la implementación continuas. Los contenedores se han vuelto una tecnología clave para la configuración e implementación de aplicaciones y microservicios. Red Hat OpenShift Container Platform es una implementación de Kubernetes, un sistema de orquestación de contenedores.

Objetivos del curso

- Demostrar su conocimiento del ecosistema de contenedores.
- Administrar los contenedores de Linux con Podman.
- Implementar contenedores en un clúster de Kubernetes con OpenShift Container Platform.
- Demostrar el diseño básico del contenedor y la capacidad de compilar imágenes de contenedores.
- Implementar una arquitectura basada en contenedores con conocimiento de contenedores, Kubernetes y OpenShift.

Destinatarios

- Administradores de sistemas
- Desarrolladores
- Líderes de TI y arquitectos de infraestructura

Requisitos previos

Los estudiantes deben cumplir con uno o más de los siguientes requisitos previos:

- Ser capaces de usar una sesión de terminal de Linux y emitir comandos del sistema operativo. Se recomienda poseer una certificación RHCSA, pero no es obligatoria.
- Tener experiencia con arquitecturas de aplicaciones web y sus correspondientes tecnologías.

Orientación sobre el entorno del aula

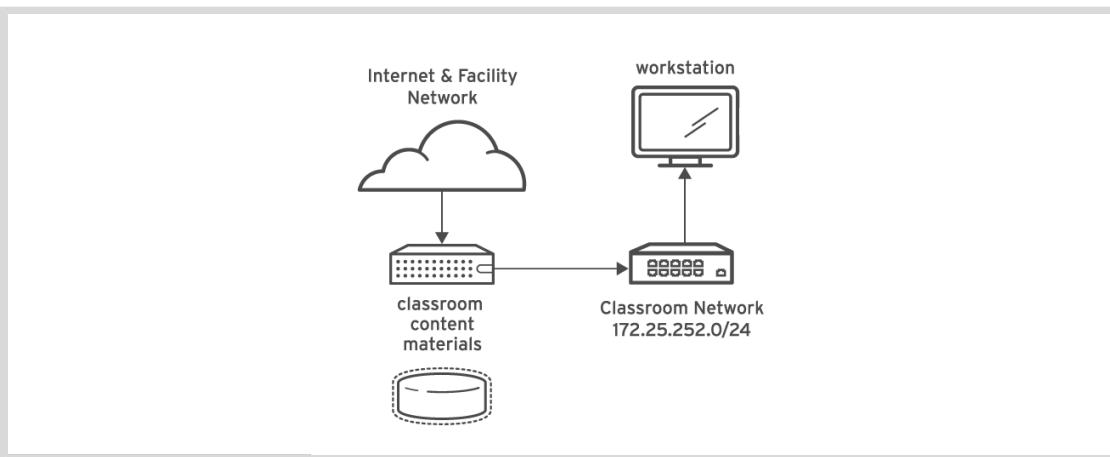


Figura 0.1: Entorno del aula

En este curso, el sistema de cómputo principal usado para las actividades prácticas de aprendizaje es **workstation**. Es una máquina virtual (VM) denominada **workstation.lab.example.com**.

Todos los sistemas de cómputo de los estudiantes tienen una cuenta de usuario estándar (**student**) con la contraseña **student**. La contraseña **root** de todos los sistemas de los estudiantes es **redhat**.

Máquinas del aula

Nombre de la máquina	Direcciones IP	Rol
content.example.com, materials.example.com, classroom.example.com	172.25.252.254, 172.25.253.254, 172.25.254.254	Servidor de utilidades del aula
workstation.lab.example.com	172.25.250.254, 172.25.252.1	Estación de trabajo gráfica del estudiante

Varios sistemas en el aula brindan servicios de soporte. Dos servidores, **content.example.com** y **materials.example.com**, son fuentes de software y materiales del trabajo de laboratorio usados en actividades prácticas. Se provee información sobre cómo usar estos servidores en las instrucciones para estas actividades.

Los estudiantes usan la máquina **workstation** para acceder a un clúster OpenShift compartido alojado externamente en AWS. Los estudiantes no tienen privilegios de administrador de clúster en el clúster, pero eso no es necesario para completar el contenido de DO180.

A los estudiantes se les aprovisiona una cuenta en un clúster compartido de OpenShift 4 cuando aprovisionan sus entornos en la interfaz de Red Hat Online Learning. La información del clúster, como el extremo (endpoint) de la API y el CLUSTER-ID, así como su nombre de usuario y contraseña, se les presenta cuando aprovisionan su entorno.

Introducción

Los estudiantes también tienen acceso a un servidor MySQL y a un servidor Nexus alojados por el clúster OpenShift o por AWS. Las actividades prácticas de este curso proporcionan instrucciones para acceder a estos servidores cuando sea necesario.

Las actividades prácticas en DO180 también requieren que los estudiantes tengan cuentas personales en dos servicios de Internet públicos y gratuitos: GitHub y Quay.io. Los estudiantes deben crear estas cuentas si aún no las tienen (consulte el Apéndice) e iniciar sesión en estos servicios antes de comenzar la clase para verificar su acceso.

Control de sus sistemas

A los estudiantes se les asignan computadoras remotas en un aula de aprendizaje en línea de Red Hat. Se accede a ellas a través de una aplicación web alojada en rol.redhat.com [<http://rol.redhat.com>]. Los estudiantes también deben iniciar sesión en este sitio usando sus credenciales de usuario del Portal de clientes de Red Hat.

Control de las máquinas virtuales

Las máquinas virtuales del entorno de su aula se controlan a través de una página web. El estado de cada máquina virtual en el aula se muestra en la página en la pestaña **Online Lab** (Trabajo de laboratorio en línea).

Estados de la máquina

Estado de la máquina virtual	Descripción
STARTING (EN INICIO)	La máquina virtual está por arrancar.
STARTED (INICIADA)	La máquina virtual se está ejecutando y está disponible (o bien, cuando arranque, pronto lo estará).
STOPPING (EN DETENCIÓN)	La máquina virtual está por apagarse.
STOPPED (DETENIDA)	La máquina virtual se ha apagado completamente. Al iniciarse, la máquina virtual arranca en el mismo estado en que se hallaba en el momento de apagarse (el disco se habrá preservado).
PUBLISHING (PUBLICADA)	Se está llevando a cabo la creación inicial de la máquina virtual.
WAITING_TO_START (EN ESPERA PARA INICIARSE)	La máquina virtual está esperando que inicien las demás máquinas virtuales.

Según el estado de una máquina, se dispone de una selección de las siguientes acciones.

Acciones de aula/máquina

Botón o acción	Descripción
PROVISION LAB (APROVISIONAR TRABAJO DE LABORATORIO)	Crea el aula de ROL. Crea todas las máquinas virtuales necesarias para el aula y las inicia. Puede tardar algunos minutos en completarse.
DELETE LAB (ELIMINAR TRABAJO DE LABORATORIO)	Elimina el aula de ROL. Destruye todas las máquinas virtuales del aula. Precaución: Se perderán los trabajos generados en los discos.
START LAB (INICIAR TRABAJO DE LABORATORIO)	Inicia todas las máquinas virtuales en el aula.
SHUTDOWN LAB (APAGAR TRABAJO DE LABORATORIO)	Detiene todas las máquinas virtuales en el aula.
OPEN CONSOLE (ABRIR CONSOLA)	Abra una nueva pestaña en el navegador y conéctese a la consola de la máquina virtual. Los estudiantes pueden iniciar sesión directamente en la máquina virtual y ejecutar los comandos. En la mayoría de los casos, los estudiantes deben iniciar sesión en la máquina virtual workstation y usar ssh para conectarse a las otras máquinas virtuales.
ACTION (ACCIÓN) → Start(Iniciar)	Inicie (power on [encender]) la máquina virtual.
ACTION (ACCIÓN) → Shutdown(Apagar)	Apague la máquina virtual correctamente y preserve el contenido del disco.
ACTION (ACCIÓN) → Power Off(Desconectar)	Fuerce el apagado de la máquina virtual y preserve el contenido del disco. Esto equivale a desenchufar una máquina física.
ACTION (ACCIÓN) → Reset(Restablecer)	Fuerce el apagado de la máquina virtual y restablezca el disco para que vuelva a su estado original. Precaución: Se perderán los trabajos generados en el disco.

Al inicio de un ejercicio, si se le indica que restablezca el nodo de una máquina virtual, haga clic en **ACTION (ACCIÓN) → Reset (Restablecer)** solo para la máquina virtual específica.

Al inicio de un ejercicio, si se le indica que restablezca todas las máquinas virtuales, haga clic en **ACTION (ACCIÓN) → Reset (Restablecer)**

Si desea que el entorno del aula vuelva a su estado original al inicio del curso, puede hacer clic en **DELETE LAB (ELIMINAR TRABAJO DE LABORATORIO)** para eliminar el entorno del aula completo. Después de eliminar el trabajo de laboratorio, haga clic en **PROVISION LAB**

(APROVISIONAR TRABAJO DE LABORATORIO) para aprovisionar un nuevo conjunto de sistemas del aula.



Advertencia

La operación **DELETE LAB** (ELIMINAR TRABAJO DE LABORATORIO) no puede deshacerse. Se perderán todos los trabajos que haya completado en el entorno del aula hasta el momento.

Temporizador de detención automática

La inscripción al aprendizaje en línea de Red Hat les da a los estudiantes derecho a una cierta cantidad de tiempo de uso del equipo. Para ahorrar tiempo asignado de la computadora, el aula de ROL tiene un temporizador de cuenta regresiva asociado, el cual apaga el entorno del aula cuando se termina el tiempo.

Para ajustar el temporizador, haga clic en **MODIFY** (MODIFICAR) para que aparezca el cuadro de diálogo **New Autostop Time** (Nuevo tiempo de detención automática). Defina la cantidad de horas y minutos hasta que el aula deba detenerse automáticamente. Tenga en cuenta que hay un tiempo máximo de diez horas. Haga clic en **ADJUST TIME** (AJUSTAR TIEMPO) para aplicar este cambio en los ajustes del temporizador.

Internacionalización

Selección de idioma por usuario

Es posible que sus usuarios prefieran usar un idioma para su entorno de escritorio distinto al predeterminado del sistema. Quizás también quieran usar una distribución del teclado o un método de entrada distintos para su cuenta.

Configuración de idioma

En el entorno de escritorio GNOME, posiblemente el usuario deba definir el idioma de su preferencia y el método de entrada la primera vez que inicie sesión. Si no es así, la manera más simple para un usuario individual de definir el idioma de su preferencia y el método de entrada es usando la aplicación Region & Language.

Puede iniciar esta aplicación de dos maneras. Puede ejecutar el comando **gnome-control-center region** desde una ventana de terminal, o bien, en la barra superior en el menú de sistema de la esquina derecha, puede seleccionar el botón de configuración (que tiene un icono de un destornillador y una llave inglesa cruzados) ubicado en la parte inferior izquierda del menú.

En la ventana que se abre, seleccione Region & Language (Región e idioma). El usuario puede hacer clic en la caja (box) **Language** (Idioma) y seleccionar el idioma de su preferencia en la lista que aparece. Esto también actualiza la configuración de **Formats** (Formatos) mediante la adopción del valor predeterminado para ese idioma. La próxima vez que inicie sesión, se efectuarán los cambios.

Estas configuraciones afectan el entorno de escritorio GNOME y todas las aplicaciones, como **gnome-terminal**, que se inician dentro de este. Sin embargo, de forma predeterminada, no se aplican a la cuenta si el acceso a ella es mediante un inicio de sesión **ssh** desde un sistema remoto o un inicio de sesión basado en texto en una consola virtual (como **tty5**).



nota

Puede hacer que su entorno de shell use la misma configuración de **LANG** que su entorno gráfico, incluso cuando inicia sesión a través de una consola virtual basada en texto o mediante **ssh**. Una manera de hacer esto es colocar un código similar al siguiente en su archivo **~/.bashrc**. Este código de ejemplo definirá el idioma empleado en un inicio de sesión en interfaz de texto de modo que coincida con el idioma actualmente definido en el entorno de escritorio GNOME del usuario.

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

Es posible que algunos idiomas, como el japonés, coreano, chino y otros con un conjunto de caracteres no latinos, no se vean correctamente en consolas virtuales basadas en texto.

Se pueden crear comandos individuales para usar otro idioma mediante la configuración de la variable **LANG** en la línea de comandos:

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

Los comandos subsiguientes se revertirán y usará el idioma de salida predeterminado del sistema. El comando **locale** se puede usar para determinar el valor actual de **LANG** y otras variables de entorno relacionadas.

Configuración del método de entrada

GNOME 3 en Red Hat Enterprise Linux 7 o posterior usa de manera automática el sistema de selección de método de entrada IBus, que permite cambiar las distribuciones del teclado y los métodos de entrada de manera rápida y sencilla.

La aplicación Region & Language (Región e idioma) también se puede usar para habilitar métodos de entrada alternativos. En la ventana de la aplicación Region & Language (Región e idioma), la caja (box) **Input Sources** (Fuentes de entrada) muestra los métodos de entrada disponibles en este momento. De forma predeterminada, es posible que **English (US)** (Inglés [EE. UU.]) sea el único método disponible. Resalte **English (US)** (Inglés [EE. UU.]) y haga clic en el icono de **Keyboard** (teclado) para ver la distribución actual del teclado.

Para agregar otro método de entrada, haga clic en el botón **+**, en la parte inferior izquierda de la ventana **Input Sources**. Se abrirá la ventana **Add an Input Source**. Seleccione su idioma y, luego, el método de entrada o la distribución del teclado de su preferencia.

Cuando haya más de un método de entrada configurado, el usuario puede alternar entre ellos rápidamente escribiendo **Super+Space** (en ocasiones denominado **Windows+Space**). También aparecerá un *indicador de estado* en la barra superior de GNOME con dos funciones: por un lado, indica el método de entrada activo; por el otro lado, funciona como un menú que puede usarse para cambiar de un método de entrada a otro o para seleccionar funciones avanzadas de métodos de entrada más complejos.

Algunos de los métodos están marcados con engranajes, que indican que tienen opciones de configuración y capacidades avanzadas. Por ejemplo, el método de entrada japonés **Japanese (Kana Kanji)** (japonés [Kana Kanji]) permite al usuario editar previamente texto en latín y usar las teclas de **Down Arrow** (flecha hacia abajo) y **Up Arrow** (flecha hacia arriba) para seleccionar los caracteres correctos que se usarán.

El indicador también puede ser de utilidad para los hablantes de inglés de Estados Unidos. Por ejemplo, dentro de **English (United States)** (Inglés [Estados Unidos]) está la configuración del teclado **English (international AltGr dead keys)** (Inglés [internacional, teclas inactivas AltGr]), que trata **AltGr** (o la tecla **Alt** derecha) en un teclado de 104/105 teclas de una PC como una tecla modificadora "Bloq Mayús secundaria" y tecla de activación de teclas inactivas para escribir caracteres adicionales. Hay otras distribuciones alternativas disponibles, como Dvorak.

**nota**

Cualquier carácter Unicode puede ingresarse en el entorno de escritorio GNOME si conoce el código Unicode del carácter. Escriba **Ctrl+Shift+U**, seguido por el código. Después de ingresar **Ctrl+Shift+U**, aparecerá una **u** subrayada que indicará que el sistema espera la entrada del código Unicode.

Por ejemplo, la letra del alfabeto griego en minúscula lambda tiene el código U +03BB y puede ingresarse escribiendo **Ctrl+Shift+U**, luego **03BB** y, por último, **Enter**.

Valores de idioma predeterminado en todo el sistema

El idioma predeterminado del sistema está definido en US English (inglés de EE. UU.), que usa la codificación UTF-8 de Unicode como su conjunto de caracteres (**en_US.utf8**), pero puede cambiarse durante o después de la instalación.

Desde la línea de comandos, el usuario **root** puede cambiar los ajustes de configuración regional de todo el sistema con el comando **localectl**. Si **localectl** se ejecuta sin argumentos, muestra los ajustes actuales de configuración regional de todo el sistema.

Para configurar el idioma predeterminado de todo el sistema, ejecute el comando **localectl set-locale LANG=locale**, donde **locale** es el valor adecuado para la variable de entorno **LANG** de la tabla "Referencia de códigos de idioma" en este capítulo. El cambio entrará en vigencia para los usuarios en su siguiente inicio de sesión y se almacena en **/etc/locale.conf**.

```
[root@host ~]# localectl set-locale LANG=fr_FR.utf8
```

En GNOME, un usuario administrativo puede cambiar esta configuración en Region & Language (Región e idioma) haciendo clic en el botón **Login Screen** (Pantalla de inicio de sesión) ubicado en la esquina superior derecha de la ventana. Al cambiar la opción de **Language** (Idioma) de la pantalla de inicio de sesión gráfico, también ajustará el valor de idioma predeterminado de todo el sistema en el archivo de configuración **/etc/locale.conf**.

**Importante**

Las consolas virtuales basadas en texto, como **tty4**, pueden mostrar una cantidad más limitada de fuentes que los terminales en una consola virtual que ejecuta un entorno gráfico, o pseudoterminales para sesiones **ssh**. Por ejemplo, los caracteres del japonés, coreano y chino posiblemente no se visualicen como se espera en una consola virtual basada en texto. Por este motivo, debe considerar el uso de inglés u otro idioma con un conjunto de caracteres latinos para los valores predeterminados para todo el sistema.

De manera similar, las consolas virtuales basadas en texto soportan una cantidad limitada de métodos de entrada; y esto se administra de manera separada desde el entorno gráfico de escritorio. Las opciones de entrada globales pueden ser configuradas a través **localectl** de la consola de texto virtual y del entorno gráfico. Para obtener más información, consulte las páginas del manual **localectl(1)** y **vconsole.conf(5)**.

Paquetes de idiomas

Paquetes de RPM especiales llamados *langpacks* instalan paquetes de idiomas que agregan soporte para idiomas específicos. Estos paquetes de idiomas usan dependencias para instalar automáticamente paquetes de RPM adicionales que contienen localizaciones, diccionarios y traducciones para otros paquetes de software en su sistema.

Use **yum list langpacks-*** para enumerar los paquetes de idiomas que están instalados y que pueden instalarse:

```
[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...
```

Para agregar soporte de idioma, instale el paquete langpacks correcto. Por ejemplo, el siguiente comando agrega soporte para francés:

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

Use **yum repoquery --whatplements** para determinar qué paquetes de RPM pueden ser instalados por un paquete de idiomas:

```
[root@host ~]# yum repoquery --whatplements 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
```

**Importante**

Los paquetes langpacks usan dependencias débiles de RPM para instalar paquetes complementarios solo cuando el paquete principal (core) que lo necesita también está instalado.

Por ejemplo, al instalar *langpacks-fr* como se muestra en los ejemplos anteriores, el paquete *mythes-fr* solo se instalará si el tesoro *mythes* también está instalado en el sistema.

Si *mythes* se instala posteriormente en ese sistema, el paquete *mythes-fr* también se instalará automáticamente debido a la débil dependencia del paquete *langpacks-fr* ya instalado.

**Referencias**

Páginas del manual: **locale(7)**, **localectl(1)**, **locale.conf(5)**, **vconsole.conf(5)**, **unicode(7)** y **utf-8(7)**.

Las conversiones entre los nombres de las distribuciones X11 del entorno de escritorio gráfico y sus nombres en **localectl** se pueden encontrar en el archivo **/usr/share/X11/xkb/rules/base.lst**.

Referencia de códigos de idioma

**nota**

Es posible que en esta tabla no se reflejen todos los paquetes de idiomas disponibles en su sistema. Use **yum info langpacks-SUFFIX** para obtener más información sobre un paquete de idiomas en particular.

Códigos de idioma

Lenguaje	Sufijo de los paquetes de idiomas	Valor \$LANG
Inglés (EE. UU.)	en	en_US.utf8
Asamés	as	as_IN.utf8
Bengalí	bn	bn_IN.utf8
Chino (Simplificado)	zh_CN	zh_CN.utf8
Chino (Tradicional)	zh_TW	zh_TW.utf8
Francés	fr	fr_FR.utf8
Alemán	de	de_DE.utf8
Guyaratí	gu	gu_IN.utf8

Lenguaje	Sufijo de los paquetes de idiomas	Valor \$LANG
Hindi	hi	hi_IN.utf8
Italiano	it	it_IT.utf8
Japonés	ja	ja_JP.utf8
Canarés	kn	kn_IN.utf8
Coreano	ko	ko_KR.utf8
Malayalam	ml	ml_IN.utf8
Maratí	mr	mr_IN.utf8
Odia	or	or_IN.utf8
Portugués (Brasil)	pt_BR	pt_BR.utf8
Punyabí	pa	pa_IN.utf8
Ruso	ru	ru_RU.utf8
Español	es	es_ES.utf8
Tamil	ta	ta_IN.utf8
Telugú	te	te_IN.utf8

capítulo 1

Introducción a la tecnología de contenedores

Meta

Describir el modo en que las aplicaciones pueden ejecutarse en contenedores organizados por Red Hat OpenShift Container Platform.

Objetivos

- Describir la diferencia entre las aplicaciones de contenedor y las implementaciones tradicionales.
- Describir los aspectos básicos de la arquitectura de contenedores.
- Describir los beneficios de organizar las aplicaciones y de OpenShift Container Platform.

Secciones

- Descripción general de la tecnología de contenedores (y cuestionario)
- Descripción general de la arquitectura de contenedores (y cuestionario)
- Descripción general de Kubernetes y OpenShift (y cuestionario)

Descripción general de la tecnología de contenedores

Objetivos

Una vez terminada esta sección, los estudiantes deberán ser capaces de describir la diferencia entre las aplicaciones de contenedor y las implementaciones tradicionales.

Aplicaciones en contenedores

Las aplicaciones de software suelen depender de otras librerías, archivos de configuración o servicios proporcionados por el entorno de tiempo de ejecución. El entorno de tiempo de ejecución tradicional para una aplicación de software es un host físico o una máquina virtual, y las dependencias de las aplicaciones se instalan como parte del host.

Por ejemplo, considere una aplicación de Python que requiere acceso a una librería compartida común que implementa el protocolo TLS. Tradicionalmente, un administrador de sistema instala el paquete requerido que proporciona la librería compartida antes de instalar la aplicación de Python.

El principal inconveniente de las aplicaciones de software implementadas tradicionalmente es que las dependencias de la aplicación están entremezcladas con el entorno de tiempo de ejecución. Una aplicación puede fallar cuando se aplican actualizaciones o parches al sistema operativo (SO) de base.

Por ejemplo, una actualización del SO de la librería compartida de TLS elimina TLS 1.0 como protocolo soportado. Esto rompe la aplicación de Python implementada porque su código indica que se use el protocolo TLS 1.0 para las solicitudes de red. Esto obliga al administrador del sistema a revertir la actualización del SO para mantener la aplicación en ejecución, lo que evita que otras aplicaciones aprovechen los beneficios del paquete actualizado. Por lo tanto, una empresa que desarrolla aplicaciones de software tradicionales puede requerir un conjunto completo de pruebas para garantizar que una actualización del SO no afecte a las aplicaciones que se ejecutan en el host.

Además, una aplicación implementada tradicionalmente debe detenerse antes de actualizar las dependencias asociadas. Para minimizar el tiempo de inactividad de las aplicaciones, las organizaciones diseñan e implementan sistemas complejos para proporcionar una alta disponibilidad de sus aplicaciones. Mantener varias aplicaciones en un mismo host a menudo se vuelve engorroso, y cualquier implementación o actualización puede llegar a descomponer una de las aplicaciones de la organización.

En la *Figura 1.1*, se describe la diferencia entre aplicaciones que se ejecutan como contenedores y aplicaciones que se ejecutan en el sistema operativo del host.

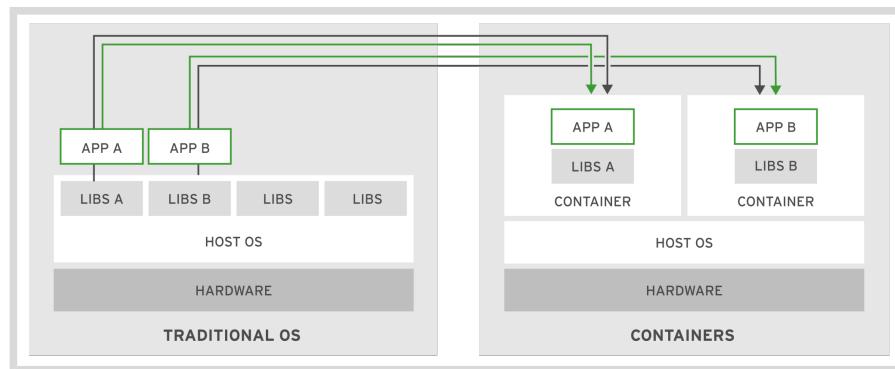


Figura 1.1: Diferencias entre contenedor y sistema operativo

Como alternativa, una aplicación de software puede implementarse con un *contenedor*. Un contenedor es un conjunto de uno o más procesos que están aislados del resto del sistema. Los contenedores proporcionan muchos de los mismos beneficios que las máquinas virtuales, como la seguridad, el almacenamiento y el aislamiento de la red. Los contenedores requieren muchos menos recursos de hardware, y se inician y se finalizan rápidamente. También aíslan las librerías y los recursos de tiempo de ejecución (como la CPU y el almacenamiento) usados por una aplicación para minimizar el impacto de una actualización de SO en el SO del host, como se describe en *Figura 1.1*.

El uso de contenedores ayuda no solo con la eficacia, la elasticidad y la reutilización de las aplicaciones alojadas, sino también con la portabilidad de las aplicaciones. *Open Container Initiative* proporciona un conjunto de estándares de la industria que definen una especificación de tiempo de ejecución de contenedor y una especificación de imagen de contenedor. La especificación de imagen define el formato del paquete de archivos y metadatos que forman una imagen de contenedor. Cuando crea una aplicación como imagen de contenedor que cumple con el estándar OCI, puede usar cualquier motor de contenedores que cumpla con OCI para ejecutar la aplicación.

Hay muchos *motores de contenedores* disponibles para administrar y ejecutar contenedores individuales, incluidos Rocket, Drawbridge, LXC, Docker y Podman. Podman está disponible en Red Hat Enterprise Linux 7.6 y versiones posteriores, y se usa en este curso para iniciar, administrar y finalizar contenedores individuales.

A continuación se presentan otras ventajas importantes del uso de contenedores:

Menor tamaño del hardware

Los contenedores usan las características internas del SO para crear un entorno aislado, donde los recursos se administren con utilidades del SO, como espacios de nombres y cgroups (grupos de control). Este enfoque minimiza la cantidad de CPU y la sobrecarga de la memoria en comparación con un hipervisor de máquina virtual. La ejecución de una aplicación en una máquina virtual es una manera de crear aislamiento del entorno en ejecución, pero requiere una capa gruesa de servicios para soportar el mismo aislamiento de menor tamaño del hardware provisto por los contenedores.

Aislamiento del entorno

Los contenedores funcionan en un entorno cerrado, donde los cambios realizados al SO del host o a otras aplicaciones no afectan al contenedor. Dado que las librerías requeridas por un contenedor son autónomas, la aplicación puede ejecutarse sin interrupción. Por ejemplo, cada aplicación puede existir en su propio contenedor con su propio conjunto de librerías. Una actualización realizada a un contenedor no afecta a otros contenedores.

Implementación rápida

Los contenedores se implementan rápidamente porque no existe la necesidad de instalar todo el sistema operativo subyacente. Generalmente, para soportar el aislamiento, se requiere una nueva instalación del SO en un host físico o una máquina virtual, y cualquier actualización simple requiere un reinicio del SO completo. Para reiniciar un contenedor, no hace falta detener los servicios del SO del host.

Implementación de varios entornos

En una situación de implementación tradicional con un solo host, cualquier diferencia entre los entornos podía interrumpir la aplicación. Sin embargo, al usar contenedores, todas las dependencias de la aplicación y la configuración del entorno están encapsuladas en la imagen del contenedor.

Reutilización

El mismo contenedor se puede reutilizar sin tener que configurar un SO completo. Por ejemplo, cada desarrollador puede usar el mismo contenedor de base de datos que proporciona un servicio de base de datos de producción para crear una base de datos de desarrollo durante el desarrollo de la aplicación. Al usar contenedores, ya no es necesario mantener servidores de base de datos separados para la producción y el desarrollo. Se usa una única imagen de contenedor para crear instancias del servicio de base de datos.

A menudo, se establece que una aplicación de software con todos sus servicios dependientes (bases de datos, mensajería, sistemas de archivos) se ejecute en un solo contenedor. Esto puede llevar a los mismos problemas asociados con las implementaciones de software tradicionales en máquinas virtuales o hosts físicos. En estas instancias, una implementación con varios contenedores puede ser más adecuada.

Además, los contenedores son un enfoque ideal cuando se usan microservicios para el desarrollo de aplicaciones. Cada servicio se encapsula en un entorno de contenedor ligero y confiable que se puede implementar en un entorno de producción o desarrollo. La colección de servicios en contenedor que requiere una aplicación se puede alojar en una sola máquina, lo que elimina la necesidad de administrar una máquina para cada servicio.

En contraste, muchas aplicaciones no son adecuadas para un entorno de contenedores. Por ejemplo, las aplicaciones que acceden a información de hardware de bajo nivel, como memoria, sistemas de archivos y dispositivos, pueden ser poco confiables debido a las limitaciones de los contenedores.



Referencias

Inicio: Open Container Initiative

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

► Cuestionario

Descripción general de la tecnología de contenedores

Elija las respuestas correctas para las siguientes preguntas:

- 1. **¿Cuáles son las dos opciones que son ejemplos de aplicaciones de software que pueden ejecutarse en un contenedor? (Elija dos opciones).**
- a. Una aplicación de Python orientada a bases de datos que accede a servicios como bases de datos MySQL, un servidor de protocolo de transferencia de archivos (FTP) y un servidor web en un solo host físicos.
 - b. Una aplicación de Java Enterprise Edition, con una base de datos Oracle, y un agente de mensajería que se ejecuta en una sola máquina virtual.
 - c. Una herramienta de monitoreo de E/S para analizar el tráfico y bloquear la transferencia de datos.
 - d. Una herramienta de aplicación de volcado de memoria capaz de tomar instantáneas de todas las cachés de la CPU de memoria con fines de depuración.
- 2. **¿Cuáles de los siguientes son los dos casos prácticos que se adaptan mejor para los contenedores? (Elija dos opciones).**
- a. Un proveedor de software necesita distribuir software que puede reutilizarse por otras empresas de manera rápida y sin errores.
 - b. Una empresa está implementando aplicaciones en un host físicos y desea mejorar su rendimiento usando contenedores.
 - c. Los desarrolladores en una empresa necesitan un entorno desecharable que imite el entorno de producción, de modo que puedan probar rápidamente el código que desarrollan.
 - d. Una empresa financiera está implementando una herramienta de análisis de riesgo con un uso intensivo de la CPU en sus propios contenedores para minimizar la cantidad de procesadores necesarios.

- 3. Una empresa está migrando sus aplicaciones Python y PHP que se ejecutan en el mismo host a una nueva arquitectura. Debido a políticas internas, ambas usan un conjunto de librerías compartidas personalizadas desde el SO, pero la actualización más reciente aplicada a estas como consecuencia de una solicitud del equipo de desarrollo de Python interrumpió la aplicación PHP. ¿Cuáles son las dos arquitecturas que proporcionarían el mejor soporte para ambas aplicaciones? (Elija dos opciones).
- a. Implementar cada aplicación en máquinas virtuales diferentes y aplicar las librerías compartidas personalizadas de forma individual en cada host de máquina virtual.
 - b. Implementar cada aplicación en contenedores diferentes y aplicar las librerías compartidas personalizadas de forma individual en cada contenedor.
 - c. Implementar cada aplicación en máquinas virtuales diferentes y aplicar las librerías compartidas personalizadas en todos los hosts de máquina virtual.
 - d. Implementar cada aplicación en contenedores diferentes y aplicar las librerías compartidas personalizadas en todos los contenedores.
- 4. ¿Cuáles son los tres tipos de aplicaciones que se pueden empaquetar como contenedores para un consumo inmediato? (Elija tres opciones).
- a. Un hipervisor para máquinas virtuales
 - b. Un software para blog, como WordPress
 - c. Una base de datos
 - d. Una herramienta de recuperación del sistema de archivos local
 - e. Un servidor web

► Solución

Descripción general de la tecnología de contenedores

Elija las respuestas correctas para las siguientes preguntas:

- 1. **¿Cuáles son las dos opciones que son ejemplos de aplicaciones de software que pueden ejecutarse en un contenedor? (Elija dos opciones).**
- a. Una aplicación de Python orientada a bases de datos que accede a servicios como bases de datos MySQL, un servidor de protocolo de transferencia de archivos (FTP) y un servidor web en un solo host físicos.
 - b. Una aplicación de Java Enterprise Edition, con una base de datos Oracle, y un agente de mensajería que se ejecuta en una sola máquina virtual.
 - c. Una herramienta de monitoreo de E/S para analizar el tráfico y bloquear la transferencia de datos.
 - d. Una herramienta de aplicación de volcado de memoria capaz de tomar instantáneas de todas las cachés de la CPU de memoria con fines de depuración.
- 2. **¿Cuáles de los siguientes son los dos casos prácticos que se adaptan mejor para los contenedores? (Elija dos opciones).**
- a. Un proveedor de software necesita distribuir software que puede reutilizarse por otras empresas de manera rápida y sin errores.
 - b. Una empresa está implementando aplicaciones en un host físicos y desea mejorar su rendimiento usando contenedores.
 - c. Los desarrolladores en una empresa necesitan un entorno desecharable que imite el entorno de producción, de modo que puedan probar rápidamente el código que desarrollan.
 - d. Una empresa financiera está implementando una herramienta de análisis de riesgo con un uso intensivo de la CPU en sus propios contenedores para minimizar la cantidad de procesadores necesarios.

- **3. Una empresa está migrando sus aplicaciones Python y PHP que se ejecutan en el mismo host a una nueva arquitectura. Debido a políticas internas, ambas usan un conjunto de librerías compartidas personalizadas desde el SO, pero la actualización más reciente aplicada a estas como consecuencia de una solicitud del equipo de desarrollo de Python interrumpió la aplicación PHP. ¿Cuáles son las dos arquitecturas que proporcionarían el mejor soporte para ambas aplicaciones? (Elija dos opciones).**
- a. Implementar cada aplicación en máquinas virtuales diferentes y aplicar las librerías compartidas personalizadas de forma individual en cada host de máquina virtual.
 - b. Implementar cada aplicación en contenedores diferentes y aplicar las librerías compartidas personalizadas de forma individual en cada contenedor.
 - c. Implementar cada aplicación en máquinas virtuales diferentes y aplicar las librerías compartidas personalizadas en todos los hosts de máquina virtual.
 - d. Implementar cada aplicación en contenedores diferentes y aplicar las librerías compartidas personalizadas en todos los contenedores.
- **4. ¿Cuáles son los tres tipos de aplicaciones que se pueden empaquetar como contenedores para un consumo inmediato? (Elija tres opciones).**
- a. Un hipervisor para máquinas virtuales
 - b. Un software para blog, como WordPress
 - c. Una base de datos
 - d. Una herramienta de recuperación del sistema de archivos local
 - e. Un servidor web

Descripción general de la arquitectura de contenedores

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Describir la arquitectura de los contenedores de Linux.
- Instalar la utilidad **podman** para administrar contenedores.

Introducción al historial de contenedores

Los contenedores han ganado popularidad rápidamente en los últimos años. Sin embargo, la tecnología detrás de los contenedores ha existido por un tiempo relativamente largo. En 2001, Linux introdujo un proyecto llamado VServer. VServer fue el primer intento de ejecutar conjuntos completos de procesos dentro de un solo servidor con un alto grado de aislamiento.

A partir de VServer, la idea de los procesos aislados evolucionó y se formalizó en torno a las siguientes características del kernel de Linux:

Espacios de nombres

El kernel puede aislar recursos específicos del sistema, que normalmente están visibles para todos los procesos, colocándolos en un espacio de nombres. Dentro del espacio de nombres, solo los procesos que son miembros de ese espacio de nombres pueden ver estos recursos. Los espacios de nombres pueden incluir recursos como interfaces de red, la lista de ID del proceso, puntos de montaje, recursos de IPC y la información del nombre de host del sistema.

Grupos de control (cgroups)

Los grupos de control partitionan conjuntos de procesos y sus elementos secundarios en grupos para administrar y limitar los recursos que consumen. Los grupos de control colocan restricciones sobre la cantidad de recursos del sistema que pueden usar los procesos. Esas restricciones impiden que un proceso use demasiados recursos en el host.

Seccomp

Desarrollado en 2005 y usado con los contenedores desde alrededor de 2014, Seccomp limita la forma en que los procesos pueden usar las llamadas al sistema. Seccomp define un perfil de seguridad para procesos, y elabora una lista de permitidos con las llamadas al sistema, los parámetros y los descriptores de archivos que pueden usar.

SELinux

SELinux (Security-Enhanced Linux) es un sistema de control de acceso obligatorio para los procesos. El kernel de Linux usa SELinux para proteger los procesos entre sí y para proteger el sistema host de sus procesos en ejecución. Los procesos se ejecutan como un tipo de SELinux confinado, que tiene acceso limitado a los recursos del sistema del host.

Todas estas innovaciones y características se centran en un concepto básico: permitir que los procesos se ejecuten de manera aislada y, al mismo tiempo, accedan a los recursos del sistema. Este concepto es la base de la tecnología de contenedores y la base para todas las implementaciones de contenedores. Hoy en día, los contenedores son procesos del kernel de Linux que usan esas características de seguridad para crear un entorno aislado. Este entorno evita que los procesos aislados usen indebidamente el sistema u otros recursos de contenedores.

Un caso de uso común de los contenedores es tener varias réplicas del mismo servicio (por ejemplo, un servidor de base de datos) en el mismo host. Cada réplica tiene recursos aislados (sistema de archivos, puertos, memoria), por lo que no es necesario que el servicio maneje el uso compartido de recursos. El aislamiento garantiza que un servicio defectuoso o dañino no pueda afectar a otros servicios o contenedores del mismo host, ni del sistema subyacente.

Descripción de la arquitectura de contenedores de Linux

Desde la perspectiva del kernel de Linux, un contenedor es un proceso con restricciones. Sin embargo, en lugar de ejecutar un solo archivo binario, un contenedor ejecuta una *imagen*. Una imagen es un paquete de sistema de archivos que contiene todas las dependencias necesarias para ejecutar un proceso: archivos del sistema de archivos, paquetes instalados, recursos disponibles, procesos en ejecución y módulos del kernel.

Así como los archivos ejecutables son la base para ejecutar procesos, las imágenes son la base para ejecutar contenedores. Los contenedores en ejecución usan una vista inmutable de la imagen, lo que permite que varios contenedores vuelvan a usar la misma imagen simultáneamente. Como las imágenes son archivos, se pueden administrar mediante sistemas de control de versiones, lo que mejora la automatización del contenedor y el aprovisionamiento de imágenes.

Las imágenes de contenedores deben estar disponibles a nivel local para que el tiempo de ejecución del contenedor las execute, pero las imágenes generalmente se almacenan y se mantienen en un *repositorio de imágenes*. Un repositorio de imágenes es solo un servicio (público o privado) donde las imágenes se pueden almacenar, buscar y recuperar. Otras características proporcionadas por los repositorios de imágenes son acceso remoto, metadatos de imagen, autorización o control de versión de imagen.

Existen muchos repositorios de imágenes diferentes disponibles, cada uno con características diferentes:

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

En este curso, se usa el registro de imágenes públicas Quay para que los estudiantes puedan trabajar con imágenes sin preocuparse porque interfieran entre sí.

Administración de contenedores con Podman

Los contenedores, las imágenes y los registros de imágenes deben poder interactuar entre sí. Por ejemplo, debe poder crear imágenes y colocarlas en registros de imágenes. También debe poder recuperar una imagen del registro de imágenes y crear un contenedor a partir de esa imagen.

Podman es una herramienta de código abierto para administrar contenedores e imágenes de contenedores, y para interactuar con registros de imágenes. Ofrece las siguientes características clave:

- Usa el formato de imagen especificado por Open Container Initiative [<https://www.opencontainers.org>] (OCI). Esas especificaciones definen un formato de imagen estándar, no patentado e impulsado por la comunidad.

capítulo 1 | Introducción a la tecnología de contenedores

- Podman almacena imágenes locales en el sistema de archivos local. Al hacerlo, se evita la arquitectura innecesaria de cliente/servidor o tener daemons en ejecución en la máquina local.
- Podman sigue los mismos patrones de comandos, como la CLI de Docker, por lo que no necesita aprender un nuevo conjunto de herramientas.
- Podman es compatible con Kubernetes. Kubernetes puede usar Podman para administrar sus contenedores.

En la actualidad, Podman solo está disponible en sistemas Linux. Para instalar Podman en Red Hat Enterprise Linux, Fedora o sistemas similares basados en RPM, ejecute **sudo yum install podman** o **sudo dnf install podman**.



Referencias

Red Hat Quay Container Registry

<https://quay.io>

Sitio de Podman

<https://podman.io/>

Open Container Initiative

<https://www.opencontainers.org>

► Cuestionario

Descripción general de la arquitectura de contenedores

Elija las respuestas correctas para las siguientes preguntas:

- ▶ 1. **¿Cuáles de las siguientes son las tres características de Linux que se usan para ejecutar contenedores? (Elija tres opciones).**
 - a. Espacios de nombres
 - b. Administración de la integridad
 - c. Linux con seguridad mejorada
 - d. Grupos de control

- ▶ 2. **¿Cuál de las siguientes opciones describe mejor una imagen de contenedor?**
 - a. Una imagen de máquina virtual a partir de la cual se creará el contenedor.
 - b. Un plano del contenedor a partir del cual se creará el contenedor.
 - c. Un entorno de tiempo de ejecución donde se ejecutará una aplicación.
 - d. El archivo de índice del contenedor que usa un registro.

- ▶ 3. **¿Cuáles son los tres componentes comunes en las implementaciones de arquitectura de contenedor? (Elija tres opciones).**
 - a. Tiempo de ejecución de contenedor
 - b. Permisos de contenedor
 - c. Imágenes de contenedores
 - d. Registros de contenedor

- ▶ 4. **¿Qué es un contenedor en relación con el kernel de Linux?**
 - a. Una máquina virtual.
 - b. Un proceso aislado con acceso regulado a los recursos.
 - c. Un conjunto de capas de sistema de archivos expuestas por UnionFS.
 - d. Un servicio externo que proporciona imágenes de contenedores.

► Solución

Descripción general de la arquitectura de contenedores

Elija las respuestas correctas para las siguientes preguntas:

- ▶ 1. **¿Cuáles de las siguientes son las tres características de Linux que se usan para ejecutar contenedores? (Elija tres opciones).**
 - a. Espacios de nombres
 - b. Administración de la integridad
 - c. Linux con seguridad mejorada
 - d. Grupos de control

- ▶ 2. **¿Cuál de las siguientes opciones describe mejor una imagen de contenedor?**
 - a. Una imagen de máquina virtual a partir de la cual se creará el contenedor.
 - b. Un plano del contenedor a partir del cual se creará el contenedor.
 - c. Un entorno de tiempo de ejecución donde se ejecutará una aplicación.
 - d. El archivo de índice del contenedor que usa un registro.

- ▶ 3. **¿Cuáles son los tres componentes comunes en las implementaciones de arquitectura de contenedor? (Elija tres opciones).**
 - a. Tiempo de ejecución de contenedor
 - b. Permisos de contenedor
 - c. Imágenes de contenedores
 - d. Registros de contenedor

- ▶ 4. **¿Qué es un contenedor en relación con el kernel de Linux?**
 - a. Una máquina virtual.
 - b. Un proceso aislado con acceso regulado a los recursos.
 - c. Un conjunto de capas de sistema de archivos expuestas por UnionFS.
 - d. Un servicio externo que proporciona imágenes de contenedores.

Resumen de Kubernetes y OpenShift

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Identificar las limitaciones de los contenedores de Linux y la necesidad de la orquestación de contenedores.
- Describir la herramienta de orquestación de contenedores de Kubernetes.
- Describir la plataforma Red Hat OpenShift Container Platform (RHOCP).

Limitaciones de los contenedores

Los contenedores proporcionan una manera fácil de empaquetar y ejecutar servicios. A medida que crece el número de contenedores administrados por una organización, el trabajo de iniciarlos manualmente aumenta exponencialmente, junto con la necesidad de responder rápidamente a las demandas externas.

Cuando se usan contenedores en un entorno de producción, las empresas a menudo requieren:

- Comunicación fácil entre una gran cantidad de servicios.
- Límites de recursos en las aplicaciones, independientemente de la cantidad de contenedores que las ejecutan.
- Responder a los picos de uso de las aplicaciones para aumentar o disminuir los contenedores en ejecución.
- Reaccionar ante el deterioro de los servicios.
- Implementar gradualmente una nueva versión para un conjunto de usuarios.

Las empresas a menudo requieren una tecnología de orquestación de contenedores porque los tiempos de ejecución de contenedores (como Podman) no abordan adecuadamente los requisitos anteriores.

Descripción general de Kubernetes

Kubernetes es un servicio de orquestación que simplifica la implementación, la administración y el escalamiento de las aplicaciones contenerizadas.

La unidad más pequeña manejable en Kubernetes es un pod. Un pod consta de uno o más contenedores con recursos de almacenamiento y dirección IP que representan una sola aplicación. Kubernetes también usa pods para organizar los contenedores dentro suyo y limitar sus recursos como una sola unidad.

Características de Kubernetes

Kubernetes ofrece las siguientes características además de una infraestructura de contenedores:

Detección de servicios y balanceo de carga

Kubernetes habilita la comunicación entre servicios al asignar una sola entrada de DNS a cada conjunto de contenedores. De esta manera, el servicio solicitante solo necesita saber el nombre de DNS del objetivo, lo que permite al clúster cambiar la ubicación y la dirección IP del contenedor sin afectar al servicio. Esto permite balancear las cargas de la solicitud entre el conjunto (pool) de contenedores que presta el servicio. Por ejemplo, Kubernetes puede dividir de manera uniforme las solicitudes entrantes a un servicio de MySQL teniendo en cuenta la disponibilidad de los pods.

Escalamiento horizontal

Las aplicaciones pueden escalarse hacia arriba o hacia abajo de forma manual o automática con la configuración establecida con la interfaz de línea de comandos de Kubernetes o con la interfaz de usuario web.

Reparación automática

Kubernetes puede usar controles de estado definidos por el usuario para monitorear los contenedores, y reiniciarlos y reprogramarlos en caso de falla.

Implementación automatizada

Kubernetes puede implementar gradualmente actualizaciones a los contenedores de su aplicación mientras comprueba su estado. Si algo sale mal durante la implementación, Kubernetes puede revertir a la iteración anterior de la implementación.

Administración de secretos y configuración

Puede administrar los ajustes de configuración y los secretos de sus aplicaciones sin modificar los contenedores. Los secretos de las aplicaciones pueden ser nombres de usuario, contraseñas y extremos de servicio; cualquier configuración que deba mantenerse privada.

Operadores

Los operadores son aplicaciones de Kubernetes empaquetadas que también llevan el conocimiento del ciclo de vida de la aplicación al clúster de Kubernetes. Las aplicaciones empaquetadas como operadores usan la API de Kubernetes para actualizar el estado del clúster ante los cambios en el estado de la aplicación.

Descripción general de OpenShift

Red Hat OpenShift Container Platform (RHOCOP) es un conjunto de componentes y servicios modulares desarrollados sobre la base de una infraestructura de contenedores de Kubernetes. RHOCOP agrega las capacidades para proporcionar una plataforma de PaaS de producción, como administración remota, multitenancy (multiinquilino), mayor seguridad, monitoreo y auditoría, administración del ciclo de vida de la aplicación e interfaces de autoservicio para desarrolladores.

Comenzando con Red Hat OpenShift v4, los hosts en un clúster de OpenShift usan todos Red Hat Enterprise Linux CoreOS como sistema operativo subyacente.

Durante este curso, los términos RHOCOP y OpenShift se usan para referirse a Red Hat OpenShift Container Platform.

Características de OpenShift

OpenShift agrega las siguientes características a un clúster de Kubernetes:

Flujo de trabajo de desarrollador integrado

RHOCOP integra un registro de contenedores incorporado, tuberías de CI/CD y S2I, una herramienta para crear artefactos desde repositorios de origen hasta imágenes de contenedores.

Rutas

Exponga fácilmente los servicios al mundo exterior.

Métricas y registro

Incluya servicio de métricas incorporado y con autoanálisis, y registro agregado.

Interfaz de usuario unificada

OpenShift trae herramientas unificadas y una interfaz de usuario para administrar todas las capacidades diferentes.



Referencias

Orquestación de contenedores de nivel de producción: Kubernetes

<https://kubernetes.io/>

OpenShift: plataforma de aplicaciones de contenedores de Red Hat, incorporada en Docker y Kubernetes

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

► Cuestionario

Descripción de Kubernetes y OpenShift

Elija las respuestas correctas para las siguientes preguntas:

- 1. **¿Cuáles de las siguientes son las tres afirmaciones correctas con respecto a las limitaciones de los contenedores? (Elija tres opciones).**
- a. Los contenedores son fáciles de organizar en grandes cantidades.
 - b. La falta de automatización aumenta el tiempo de respuesta a los problemas.
 - c. Los contenedores no administran las fallas de aplicaciones dentro de ellos.
 - d. Los contenedores no tienen balanceo de carga.
 - e. Los contenedores son aplicaciones empaquetadas muy aisladas.
- 2. **¿Cuáles de las siguientes son las dos afirmaciones correctas con respecto a Kubernetes? (Elija dos opciones).**
- a. Kubernetes es un contenedor.
 - b. Kubernetes solo puede usar contenedores Docker.
 - c. Kubernetes es un sistema de orquestación de contenedores.
 - d. Kubernetes simplifica la administración, la implementación y el escalamiento de las aplicaciones en contenedores.
 - e. Las aplicaciones administradas en un clúster de Kubernetes son más difíciles de mantener.
- 3. **¿Cuáles de las siguientes son las tres afirmaciones verdaderas con respecto a Red Hat OpenShift v4? (Elija tres opciones).**
- a. OpenShift proporciona características adicionales a una infraestructura de Kubernetes.
 - b. Kubernetes y OpenShift son mutuamente excluyentes.
 - c. Los hosts de OpenShift usan Red Hat Enterprise Linux como sistema operativo base.
 - d. OpenShift simplifica el desarrollo al incorporar tecnología de origen a imagen y tuberías de CI/CD.
 - e. OpenShift simplifica el enrutamiento y el balanceo de carga.
- 4. **¿Qué funciones ofrece OpenShift que amplían las capacidades de Kubernetes? (Elija dos opciones).**
- a. Operadores y Operator Framework.
 - b. Rutas para exponer los servicios al mundo exterior.
 - c. Un flujo de trabajo de desarrollo integrado.
 - d. Comprobaciones de estado y recuperación automática.

► Solución

Descripción de Kubernetes y OpenShift

Elija las respuestas correctas para las siguientes preguntas:

► 1. **¿Cuáles de las siguientes son las tres afirmaciones correctas con respecto a las limitaciones de los contenedores? (Elija tres opciones).**

- a. Los contenedores son fáciles de organizar en grandes cantidades.
- b. La falta de automatización aumenta el tiempo de respuesta a los problemas.
- c. Los contenedores no administran las fallas de aplicaciones dentro de ellos.
- d. Los contenedores no tienen balanceo de carga.
- e. Los contenedores son aplicaciones empaquetadas muy aisladas.

► 2. **¿Cuáles de las siguientes son las dos afirmaciones correctas con respecto a Kubernetes? (Elija dos opciones).**

- a. Kubernetes es un contenedor.
- b. Kubernetes solo puede usar contenedores Docker.
- c. Kubernetes es un sistema de orquestación de contenedores.
- d. Kubernetes simplifica la administración, la implementación y el escalamiento de las aplicaciones en contenedores.
- e. Las aplicaciones administradas en un clúster de Kubernetes son más difíciles de mantener.

► 3. **¿Cuáles de las siguientes son las tres afirmaciones verdaderas con respecto a Red Hat OpenShift v4? (Elija tres opciones).**

- a. OpenShift proporciona características adicionales a una infraestructura de Kubernetes.
- b. Kubernetes y OpenShift son mutuamente excluyentes.
- c. Los hosts de OpenShift usan Red Hat Enterprise Linux como sistema operativo base.
- d. OpenShift simplifica el desarrollo al incorporar tecnología de origen a imagen y tuberías de CI/CD.
- e. OpenShift simplifica el enrutamiento y el balanceo de carga.

► 4. **¿Qué funciones ofrece OpenShift que amplían las capacidades de Kubernetes? (Elija dos opciones).**

- a. Operadores y Operator Framework.
- b. Rutas para exponer los servicios al mundo exterior.
- c. Un flujo de trabajo de desarrollo integrado.
- d. Comprobaciones de estado y recuperación automática.

► Ejercicio Guiado

Configuración del entorno del aula

En este ejercicio, configurará la estación de trabajo para acceder a toda la infraestructura usada por este curso.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Configurar su estación de trabajo para acceder a un clúster de OpenShift, un registro de imágenes de contenedores y un repositorio Git usado en todo el curso.
- Bifurcar el repositorio de las aplicaciones de muestra de este curso a su cuenta de GitHub personal.
- Clonar el repositorio de las aplicaciones de muestra de este curso desde su cuenta de GitHub personal a su máquina virtual **workstation**.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de contar con lo siguiente:

- Acceso al curso DO180 en el entorno de aprendizaje en línea de Red Hat Training.
- Los parámetros de conexión y una cuenta de usuario de desarrollador para acceder a un clúster OpenShift administrado por Red Hat Training.
- Una cuenta personal y gratuita de GitHub. Si necesita registrarse en GitHub, consulte las instrucciones en *Apéndice B, Creación de una cuenta GitHub*.
- Una cuenta personal y gratuita de Quay.io. Si necesita registrarse en Quay.io, consulte las instrucciones en *Apéndice C, Creación de una cuenta Quay*.

► 1. Antes de iniciar cualquier ejercicio, debe configurar su máquina virtual **workstation**.

Para los pasos siguientes, use los valores que le proporciona el entorno de aprendizaje en línea de la Capacitación Red Hat al aprovisionar su entorno del trabajo de laboratorio en línea:

The screenshot shows the 'Lab Environment' tab selected in the top navigation bar. Below it, a box contains instructions about creating and deleting the lab environment. At the bottom are 'DELETE' and 'STOP' buttons, and a help icon. A red box highlights the 'OpenShift Details' section, which lists connection parameters:

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	

Below this is a table showing two virtual machines: 'workstation' and 'classroom', both listed as 'active'. Each has an 'ACTION' dropdown and an 'OPEN CONSOLE' button.

Abra un terminal en la máquina virtual **workstation** y ejecute el siguiente comando. Responda a sus preguntas interactivas para configurar su estación de trabajo antes de iniciar cualquier otro ejercicio de este curso.

Si comete un error, puede interrumpir el comando en cualquier momento con **Ctrl+C** y empezar de nuevo.

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

- El comando **lab-configure** comienza mostrando una serie de mensajes interactivos, e intentará encontrar algunos valores predeterminados razonables para algunos de ellos.

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...

- La URL de la API maestra de su clúster OpenShift. Escriba la URL como una sola línea, sin espacios ni saltos de línea. Capacitación Red Hat le proporciona esta información cuando aprovisiona su entorno del trabajo de laboratorio. Necesita esta información para iniciar sesión en el clúster y también para implementar aplicaciones en contenedores.
- Su nombre de usuario y contraseña de desarrollador de OpenShift. Capacitación Red Hat le proporciona esta información cuando aprovisiona su entorno del

trabajo de laboratorio. Debe usar este nombre de usuario y contraseña para iniciar sesión en OpenShift. También usará su nombre de usuario como parte de los identificadores, por ejemplo, nombres de host de ruta y nombres de proyecto, para evitar colisiones con identificadores de otros estudiantes que comparten el mismo clúster de OpenShift con usted.

- ④ ⑤ Sus nombres de cuentas de GitHub y Quay.io personales. Necesita cuentas gratuitas y válidas en estos servicios en línea para realizar los ejercicios de este curso. Si nunca ha usado ninguno de estos servicios en línea, consulte *Apéndice B, Creación de una cuenta GitHub* y *Apéndice C, Creación de una cuenta Quay* para obtener instrucciones sobre cómo registrarse.



nota

Si usa la autenticación de dos factores con su cuenta de GitHub, es posible que desee crear un token de acceso personal para su uso desde la máquina virtual **workstation** durante el curso. Consulte la siguiente documentación sobre cómo configurar un acceso personal en su cuenta: Creación de un token de acceso personal para la línea de comandos [<https://help.github.com/en/articles/creating-a-personal-access-token-for-the-command-line>]

- 1.2. El comando **lab-configure** imprime toda la información ingresada e intenta conectarse a su clúster de 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...
```

- 1.3. Si **lab-configure** encuentra algún problema, muestra un mensaje de error y sale. Deberá verificar la información y ejecutar el comando **lab-configure** nuevamente. En la siguiente lista, se muestra un ejemplo de un error de verificación:

```
...output omitted...

Verifying your Master API URL...

ERROR:
Cannot connect to an OpenShift 4.5 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.
No changes made to your lab configuration.
```

- 1.4. Si todo es correcto hasta el momento, el **lab-configure** intenta acceder a las cuentas públicas GitHub y Quay.io:

```
...output omitted...

Verifying your GitHub account name...

Verifying your Quay.io account name...

...output omitted...
```

15. Nuevamente, si **lab-configure** encuentra algún problema, muestra un mensaje de error y sale. Deberá verificar la información y ejecutar el comando **lab-configure** nuevamente. En la siguiente lista, se muestra un ejemplo de un error de verificación:

```
...output omitted...

Verifying your GitHub account name...

ERROR:
Cannot find a GitHub account named: invalidusername.
No changes made to your lab configuration.
```

16. Por último, el comando **lab-configure** verifica que el clúster de OpenShift informe sobre el dominio comodín esperado.

```
...output omitted...

Verifying your cluster configuration...

...output omitted...
```

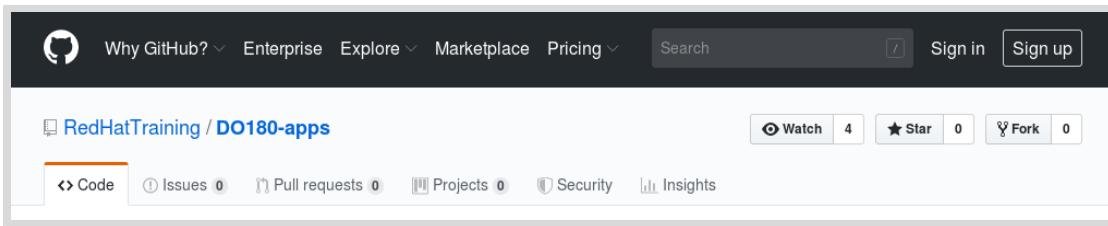
17. Si se superan todas las comprobaciones, el comando **lab-configure** guarda su configuración:

```
...output omitted...

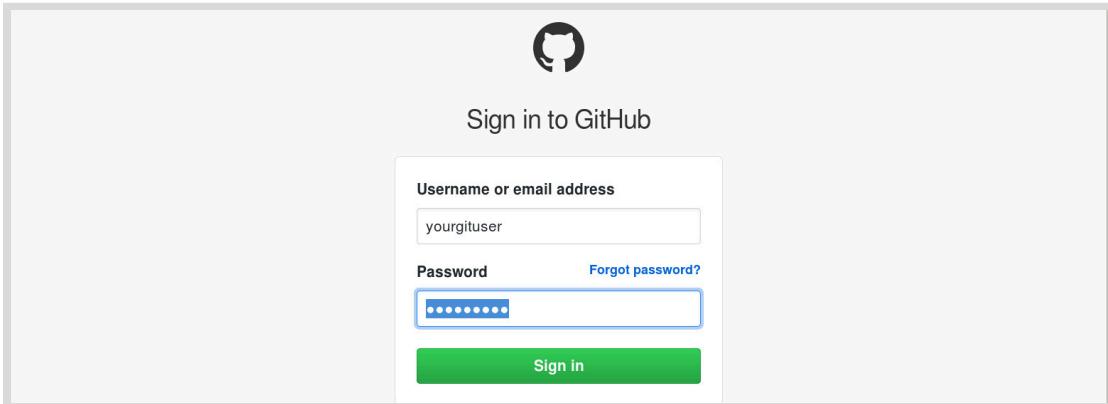
Saving your lab configuration file...

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

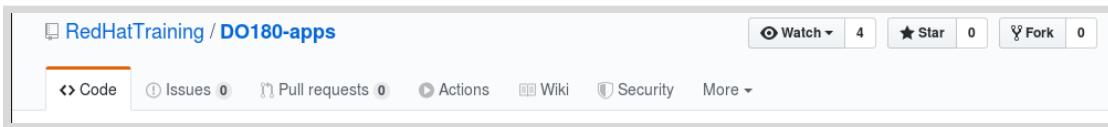
18. Si no hubo errores al guardar su configuración, está casi listo para iniciar cualquiera de los ejercicios de este curso. Si hubo errores, no intente iniciar ningún ejercicio hasta que pueda ejecutar el comando **lab-configure** correctamente.
- ▶ 2. Antes de comenzar cualquier ejercicio, necesita bifurcar las aplicaciones de muestra de este curso en su cuenta de GitHub personal. Realice los siguientes pasos:
 - 2.1. Abra un navegador web y diríjase a <https://github.com/RedHatTraining/DO180-apps>. Si no inició sesión en GitHub, haga clic en **Sign in** (Inicio de sesión) en la esquina superior derecha.



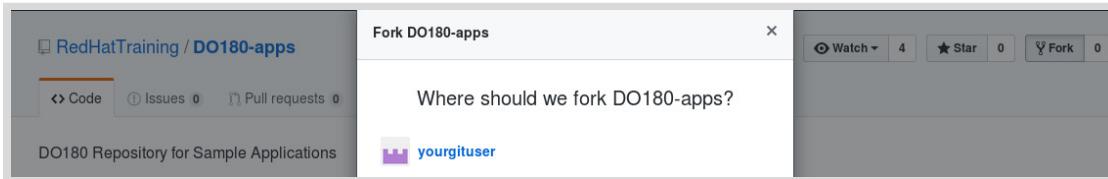
- 2.2. Inicie sesión en GitHub con su nombre de usuario personal y su contraseña.



- 2.3. Regrese al repositorio **RedHatTraining/DO180-apps** y haga clic en **Fork** (Bifurcar) en la esquina superior derecha.



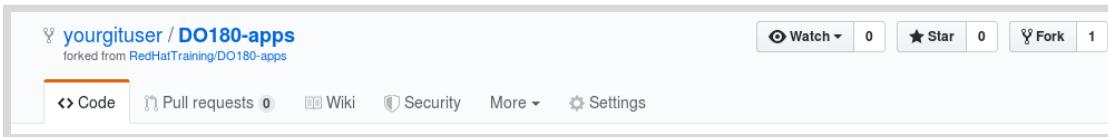
- 2.4. En la ventana **Fork DO180-apps**, haga clic en **yourgituser** para seleccionar su proyecto de GitHub personal.



Importante

Aunque es posible cambiar el nombre de la bifurcación personal del repositorio de <https://github.com/RedHatTraining/DO180-apps>, los scripts de calificación, los scripts auxiliares y la salida de ejemplo de este curso suponen que conserva el nombre **DO180-apps** cuando bifurca el repositorio.

- 2.5. Después de unos minutos, en la interfaz web de GitHub, se muestra su nuevo repositorio **yourgituser/DO180-apps**.



- 3. Antes de comenzar cualquier ejercicio, también debe clonar las aplicaciones de muestra de este curso desde su cuenta de GitHub personal hacia su máquina virtual **workstation**. Realice los siguientes pasos:

- 3.1. Ejecute el siguiente comando para clonar el repositorio de las aplicaciones de muestra de este curso. Reemplace **yourgituser** con el nombre de su cuenta de GitHub personal:

```
[student@workstation ~]$ git clone https://github.com/yourgituser/D0180-apps
Cloning into 'D0180-apps'...
...output omitted...
```

- 3.2. Verifique que **/home/student/D0180-apps** sea un repositorio Git:

```
[student@workstation ~]$ cd D0180-apps
[student@workstation D0180-apps]$ git status
# On branch master
nothing to commit, working directory clean
```

- 3.3. Verifique que **/home/student/D0180-apps** contenga las aplicaciones de muestra de este curso y vuelva a cambiar a la carpeta de inicio del usuario **student**.

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

- 4. Ahora que tiene un clon local del repositorio **D0180-apps** en su máquina virtual **workstation**, y ha ejecutado el comando **lab-configure** correctamente, está listo para iniciar los ejercicios de este curso.

Durante el curso, todos los ejercicios que compilan aplicaciones desde el origen se inician desde la bifurcación **master** del repositorio de Git **D0180-apps**. Los ejercicios que realizan cambios en el código fuente requieren la creación de nuevas bifurcaciones para alojar los cambios, de modo que la bifurcación **maestra** contenga siempre un buen punto de partida conocido. Si, por algún motivo, necesita pausar o reiniciar un ejercicio, y necesita guardar o descartar los cambios que realiza en sus bifurcaciones Git, consulte *Apéndice D, Comandos Git útiles*.

Esto concluye el ejercicio guiado.

Resumen

En este capítulo, aprendió lo siguiente:

- Los contenedores son un tiempo de ejecución de aplicaciones aislado, creado con muy poca sobrecarga.
- Una imagen de contenedor empaqueta una aplicación con todas sus dependencias, lo que facilita la ejecución de la aplicación en diferentes entornos.
- Las aplicaciones como Podman crean contenedores con funciones del kernel de Linux estándar.
- Los registros de imágenes de contenedores son el mecanismo preferido para distribuir imágenes de contenedores a varios usuarios y hosts.
- OpenShift orquesta aplicaciones compuestas por varios contenedores con Kubernetes.
- Kubernetes administra el balanceo de carga, la alta disponibilidad y el almacenamiento persistente para aplicaciones en contenedores.
- OpenShift agrega a Kubernetes multitenancy (multiinquilino), seguridad, facilidad de uso y características de integración continua/desarrollo continuo.
- Las rutas de OpenShift permiten el acceso externo a aplicaciones contenerizadas de manera que se puedan administrar.

capítulo 2

Creación de servicios contenerizados

Meta

Aprovisionar un servicio con la tecnología de contenedores.

Objetivos

- Crear un servidor de base de datos a partir de una imagen de contenedor.

Secciones

- Aprovisionamiento de un servidor de base de datos en contenedores (y ejercicio guiado)

Trabajo de laboratorio

- Creación de servicios en contenedores

Aprovisionamiento de servicios contenerizados

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Buscar y obtener imágenes de contenedor con Podman.
- Ejecutar y configurar contenedores a nivel local.
- Usar Red Hat Container Catalog.

Obtención de imágenes de contenedores con Podman

Las aplicaciones pueden ejecutarse dentro de contenedores para proporcionarles un entorno de ejecución aislado y controlado. Ejecutar una aplicación contenedora, es decir, ejecutar una aplicación dentro de un contenedor, requiere una imagen de contenedor y un conjunto de sistema de archivos que proporcione todos los archivos, las librerías y las dependencias que la aplicación necesita para ejecutarse. Las imágenes de contenedores se pueden encontrar en los registros de imágenes: servicios que permiten a los usuarios buscar y recuperar imágenes de contenedores. Los usuarios de Podman pueden usar el subcomando **search** para encontrar imágenes disponibles en registros remotos o locales:

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

Una vez que haya encontrado una imagen, puede usar Podman para descargarla. Al usar el subcomando **pull**, Podman obtiene la imagen y la guarda localmente para su uso futuro:

```
[student@workstation ~]$ sudo 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
```

Las imágenes de los contenedores se nombran según la siguiente sintaxis:

registry_name/user_name/image_name:tag

- Primero **registry_name**, el nombre del registro que almacena la imagen. Suele ser el FQDN del registro.
- **user_name** representa al usuario o la organización a la que pertenece la imagen.

- **image_name** debe ser único en el espacio de nombres de usuario.
- **tag** identifica la versión de la imagen. Si el nombre de la imagen no incluye etiqueta de imagen, se supone que es la **última**.



nota

La instalación de Podman de esta aula usa varios registros disponibles públicamente, como Quay.io y Red Hat Container Catalog.

Después de la recuperación, Podman almacena las imágenes localmente y se las puede enumerar con el subcomando **images**:

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

Ejecución de contenedores

El comando **podman run** ejecuta un contenedor localmente basado en una imagen. Como mínimo, el comando requiere que el nombre de la imagen se ejecute en el contenedor.

La imagen del contenedor especifica un proceso que comienza dentro del contenedor, conocido como *punto de entrada*. El comando **podman run** usa todos los parámetros después del nombre de la imagen como el comando de punto de entrada para el contenedor. En el siguiente ejemplo, se inicia un contenedor desde una imagen de Red Hat Enterprise Linux. Establece el punto de entrada para este contenedor en el comando **echo "Hello world"**.

```
[student@workstation ~]$ sudo podman run ubi7/ubi:7.7 echo 'Hello!'
Hello world
```

Para iniciar una imagen de contenedor como proceso en segundo plano, pase la opción **-d** al comando **podman run**:

```
[student@workstation ~]$ sudo podman run -d rhel7/httpd-24-rhel7:2.4-36.8
ff4ec6d74e9b2a7b55c49f138e56f8bc46fe2a09c23093664fea7febcb3dfa1b2
[student@workstation ~]$ sudo podman inspect -l \
> -f "{{.NetworkSettings.IPAddress}}"
10.88.0.68
[student@workstation ~]$ curl http://10.88.0.68:8080
...output omitted...
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
...output omitted...
<title>
Test Page for the Apache HTTP Server on Red Hat Enterprise Linux
</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<style type="text/css">
...output omitted...
```

En el ejemplo anterior, se ejecutó un servidor HTTP Apache contenedorizado en segundo plano. Entonces, el ejemplo usa el comando **podman inspect** para recuperar la dirección IP interna del contenedor de los metadatos del contenedor. Por último, usa la dirección IP para obtener la página raíz del servidor HTTP Apache. Esta respuesta prueba que el contenedor todavía está en funcionamiento después del comando **podman run**.

**nota**

La mayoría de los subcomandos de Podman aceptan el indicador **-l** (**l** para el último) como reemplazo para el identificador del contenedor. Este indicador aplica el comando al último contenedor usado en cualquier comando de Podman.

**nota**

Si la imagen que se ejecutará no está disponible localmente al usar el comando **podman run**, Podman usa automáticamente **pull** para descargar la imagen.

Cuando se hace referencia al contenedor, Podman reconoce un contenedor con el nombre del contenedor o con el ID del contenedor generado. Use la opción **--name** para establecer el nombre del contenedor al ejecutar el contenedor con Podman. Los nombres de los contenedores deben ser únicos. Si el comando **podman run** no incluye ningún nombre de contenedor, Podman genera un nombre aleatorio único.

Si las imágenes deben interactuar con el usuario con la entrada de la consola, Podman puede redirigir las entradas y salidas del contenedor a la consola. El subcomando **run** requiere los indicadores **-t** y **-i** (o, en definitiva, el indicador **-it**) para habilitar la interactividad.

**nota**

Muchos indicadores de Podman también tienen una forma larga alternativa; algunos de estos se explican a continuación.

- **-t** es equivalente a **--tty**, lo que significa que se debe asignar un **pseudo-tty** (pseudoterminal) para el contenedor.
- **-i** es lo mismo que **--interactive**. Cuando se usa, la entrada estándar se mantiene abierta en el contenedor.
- **-d**, o su forma larga **--detach**, significa que el contenedor se ejecuta en segundo plano (independiente). A continuación, Podman imprime el identificador del contenedor.

Consulte la documentación de Podman para ver la lista completa de indicadores.

El siguiente ejemplo inicia un terminal de Bash *dentro* del contenedor, y ejecuta interactivamente algunos comandos en él:

```
[student@workstation ~]$ sudo podman run -it ubi7/ubi:7.7 /bin/bash
bash-4.2# ls
...output omitted...
bash-4.2# whoami
root
bash-4.2# exit
exit
[student@workstation ~]$
```

Algunos contenedores necesitan o pueden usar parámetros externos provistos en el inicio. El enfoque más común para proporcionar y consumir esos parámetros es a través de variables de entorno. Podman puede insertar variables de entorno en contenedores al inicio agregando el indicador **-e** al subcomando **run**:

```
[student@workstation ~]$ sudo podman run -e GREET=Hello -e NAME=RedHat \
> rhel7:7.5 printenv GREET NAME
Hello
RedHat
[student@workstation ~]$
```

En el ejemplo anterior, se inicia un contenedor de imágenes de RHEL que imprime las dos variables de entorno proporcionadas como parámetros. Otro caso de uso para las variables de entorno es la configuración de credenciales en un servidor de base de datos de MySQL:

```
[root@workstation ~]# sudo podman run --name mysql-custom \
> -e MYSQL_USER=redhat -e MYSQL_PASSWORD=r3dh4t \
> -d rhmap47/mysql:5.5
```

Uso de Red Hat Container Catalog

Red Hat mantiene su repositorio de imágenes de contenedores optimizadas. El uso de este repositorio proporciona a los clientes una capa de protección y confiabilidad contra vulnerabilidades conocidas que podrían generarse por imágenes no sometidas a pruebas. El comando estándar **podman** es compatible con Red Hat Container Catalog. Red Hat Container Catalog proporciona una interfaz fácil de usar para buscar y explorar imágenes de contenedores del repositorio de Red Hat.

Container Catalog también funciona como una interfaz única que proporciona acceso a diferentes aspectos de todas las imágenes de contenedores disponibles en el repositorio. Es útil para determinar la mejor versión entre varias versiones de imágenes de contenedores según los niveles del índice de estado. El nivel de índice de estado indica qué tan actual es una imagen y si contiene las actualizaciones de seguridad más recientes.

Container Catalog también proporciona acceso a la documentación de erratas de una imagen. Describe las correcciones de errores y mejoras más recientes de cada actualización. También sugiere la mejor técnica para extraer una imagen en cada sistema operativo.

En las siguientes imágenes, se destacan algunas de las características de Red Hat Container Catalog.

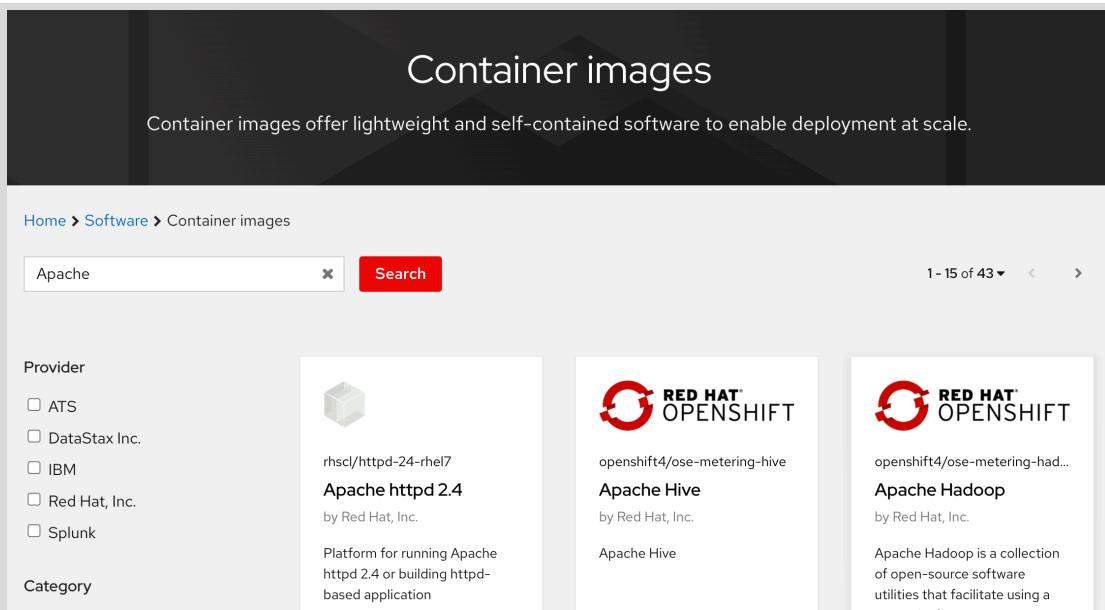


Figura 2.1: Página de búsqueda de Red Hat Container Catalog

Como se muestra arriba, buscar **Apache** en el cuadro de búsqueda de Container Catalog abre una lista sugerida de productos y repositorios de imágenes que coinciden con el patrón de búsqueda. Para acceder a la página de imagen **Apache httpd 2.4**, seleccione **rhsc1/httpd-24-rhel7** de la lista sugerida.

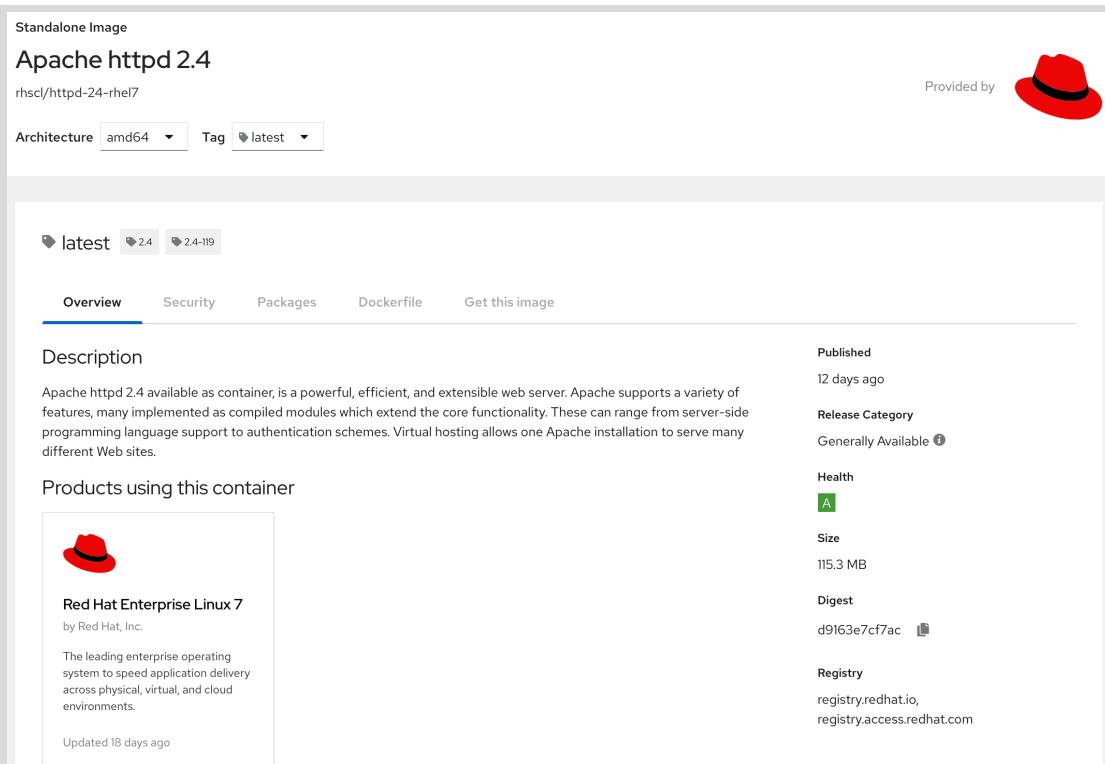


Figura 2.2: Página de imagen de resumen de Apache httpd 2.4 (rhsc1/httpd-24-rhel7)

En el panel *Apache httpd 2.4*, se muestran detalles de la imagen y varias pestañas. En esta página, se indica que Red Hat mantiene el repositorio de imágenes. En la pestaña *Overview* (Descripción general), se muestran otros detalles:

capítulo 2 | Creación de servicios contenerizados

- *Description* (Descripción): Un resumen de las capacidades de la imagen.
- *Productos que usan este contenedor*: indica que Red Hat Enterprise Linux usa este repositorio de imágenes.
- *Most Recent Tag* (Etiqueta más reciente): Cuándo la imagen recibió su actualización más reciente, la última etiqueta aplicada a la imagen, el estado de la imagen, y mucho más.

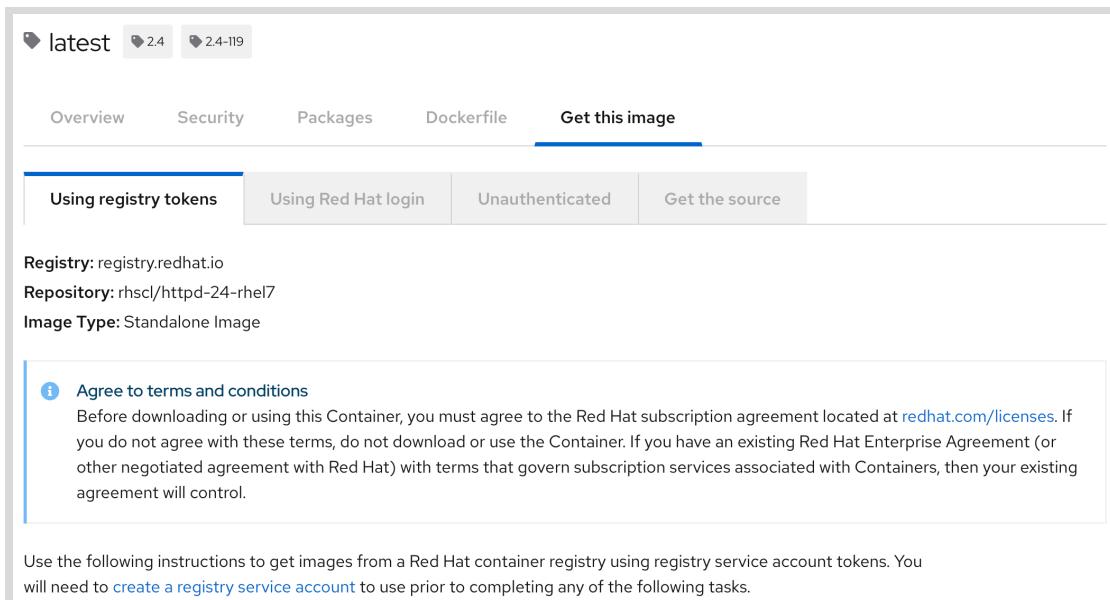


Figura 2.3: Página de imagen más reciente de Apache httpd 2.4 (rhsc1/httpd-24-rhel7)

En la pestaña *Get this image* (Obtener esta imagen), se brinda el procedimiento para obtener la versión más reciente de la imagen. En la página, se proporcionan diferentes opciones para recuperar la imagen. Elija su procedimiento preferido en las pestañas, y la página proporciona las instrucciones adecuadas para recuperar la imagen.



Referencias

Red Hat Container Catalog

<https://registry.redhat.io>

Sitio web de Quay.io

<https://quay.io>

► Ejercicio Guiado

Creación de una instancia de base de datos MySQL

En este ejercicio, iniciará una base de datos MySQL dentro de un contenedor y, luego, creará y completará la base de datos.

Resultados

Debería poder iniciar una base de datos desde una imagen de contenedor y almacenar información dentro de la base de datos.

Andes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

- 1. Cree una instancia de contenedor MySQL.

- 1.1. Inicie un contenedor de la imagen MySQL de Red Hat Software Collections Library.

```
[student@workstation ~]$ sudo podman run --name mysql-basic \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> -d rh scl/mysql-57-rhel7:5.7-3.14
Trying to pull ...output omitted...
Copying blob sha256:e373541...output omitted...
69.66 MB / 69.66 MB [=====] 8s
Copying blob sha256:c5d2e94...output omitted...
1.20 KB / 1.20 KB [=====] 0s
Copying blob sha256:b3949ae...output omitted...
62.03 MB / 62.03 MB [=====] 8s
Writing manifest to image destination
Storing signatures
92eaa6b67da0475745b2beffa7e0895391ab34ab3bf1ded99363bb09279a24a0
```

Este comando descarga la imagen de contenedor MySQL con la etiqueta **5.7-3.14** y, luego, inicia una imagen basada en contenedores. Crea una base de datos nombrada **items**, que es propiedad de un usuario nombrado **user1** con **mypa55** como contraseña. La contraseña del administrador de la base de datos se configura con **r00tpa55** y el contenedor se ejecuta en segundo plano.

- 1.2. Compruebe que el contenedor se haya iniciado sin errores.

```
[student@workstation ~]$ sudo podman ps --format "{{.ID}} {{.Image}} {{.Names}}"
92eaa6b67da0 registry.access.redhat.com/rh scl/mysql-57-rhel7:5.7-3.14 mysql-basic
```

- 2. Acceda al espacio aislado del contenedor mediante la ejecución del siguiente comando:

```
[student@workstation ~]$ sudo podman exec -it mysql-basic /bin/bash  
bash-4.2$
```

Este comando inicia una shell de Bash, que se ejecuta como el usuario **mysql** dentro del contenedor **MySQL**.

- 3. Agregue datos a la base de datos.

- 3.1. Conéctese a MySQL como el usuario administrador de bases de datos (root).

Ejecute el siguiente comando desde el terminal del contenedor para conectarse con la base de datos:

```
bash-4.2$ mysql -uroot  
Welcome to the MySQL monitor. Commands end with ; or \g.  
...output omitted...  
mysql>
```

El comando **mysql** abre el prompt interactivo de la base de datos MySQL. Ejecute el siguiente comando para determinar la disponibilidad de la base de datos:

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

- 3.2. Cree una nueva tabla en la base de datos **items** (elementos). Ejecute el siguiente comando para acceder a la base de datos.

```
mysql> use items;  
Database changed
```

- 3.3. Cree una tabla denominada **Projects** (Proyectos) en la base de datos **items** (ítems).

```
mysql> CREATE TABLE Projects (id int(11) NOT NULL,  
-> name varchar(255) DEFAULT NULL,  
-> code varchar(255) DEFAULT NULL,  
-> PRIMARY KEY (id));  
Query OK, 0 rows affected (0.01 sec)
```

Opcionalmente, puede usar el archivo **~/DO180/solutions/container-create/create_table.txt** para copiar y pegar la instrucción **CREATE TABLE** (CREAR TABLA) de MySQL como se indica anteriormente.

- 3.4. Use el comando **show tables** para verificar que se haya creado la tabla.

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

- 3.5. Use el comando **insert** para insertar una fila en la tabla.

```
mysql> insert into Projects (id, name, code) values (1, 'DevOps', 'D0180');
Query OK, 1 row affected (0.02 sec)
```

- 3.6. Use el comando **select** para verificar que la información del proyecto se haya agregado a la tabla.

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

- 3.7. Salga del prompt de MySQL y del contenedor de MySQL:

```
mysql> exit
Bye
bash-4.2$ exit
exit
```

Finalizar

En **workstation**, ejecute el script **lab container-create finish** para terminar este trabajo de laboratorio.

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

Esto concluye el ejercicio.

► Trabajo de laboratorio

Creación de servicios en contenedores

Lista de verificación de rendimiento

En este trabajo de laboratorio, creará un contenedor del servidor HTTP de Apache con una página de bienvenida personalizada.

Resultados

Deberá ser capaz de iniciar y personalizar un contenedor con una imagen de contenedor.

Antes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

- Inicie un contenedor nombrado **httpd-basic** en segundo plano y reenvíe el puerto 8080 al puerto 80 en el contenedor. Use la imagen de contenedor **redhattraining/httpd-parent** con la etiqueta **2.4**.



nota

Use la opción **-p 8080:80** con el comando **sudo podman run** para reenviar el puerto.

Este comando inicia el servidor HTTP de Apache en segundo plano y vuelve al prompt de Bash.

- Pruebe el contenedor **httpd-basic**.

Desde **workstation**, intente acceder a `http://localhost:8080` con cualquier navegador web.

Se muestra un mensaje **Hello from the httpd-parent container!** (Hola desde el contenedor httpd-parent), que es la página **index.html** del contenedor del servidor HTTP Apache que se ejecuta en **workstation**.

- Personalice el contenedor **httpd-basic** para mostrar **Hello World** (Hola, mundo) como el mensaje. El mensaje del contenedor se almacena en el archivo **/var/www/html/index.html**.

- Inicie una sesión de Bash dentro del contenedor.
- En la sesión de Bash, verifique el archivo **index.html** en el directorio **/var/www/html** con el comando **ls -la**.
- Cambie el archivo **index.html** para contener el texto **Hello World** (Hola, mundo), que reemplaza todo el contenido existente.

- 3.4. Intente acceder a `http://localhost:8080` nuevamente y verifique que se haya actualizado la página web.

Evaluación

Evalúe su trabajo ejecutando el comando `lab container-review grade` en su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el script `lab container-review finish` para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

► Solución

Creación de servicios en contenedores

Lista de verificación de rendimiento

En este trabajo de laboratorio, creará un contenedor del servidor HTTP de Apache con una página de bienvenida personalizada.

Resultados

Deberá ser capaz de iniciar y personalizar un contenedor con una imagen de contenedor.

Antes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

- Inicie un contenedor nombrado **httpd-basic** en segundo plano y reenvíe el puerto 8080 al puerto 80 en el contenedor. Use la imagen de contenedor **redhattraining/httpd-parent** con la etiqueta **2.4**.



nota

Use la opción **-p 8080:80** con el comando **sudo podman run** para reenviar el puerto.

Ejecute el siguiente comando:

```
[student@workstation ~]$ sudo podman run -d -p 8080:80 \
> --name httpd-basic redhattraining/httpd-parent:2.4
...output omitted...
Copying blob sha256:743f2d6...output omitted...
21.45 MB / 21.45 MB [=====] 1s
Copying blob sha256:c92eb69...output omitted...
155 B / 155 B [=====] 0s
Copying blob sha256:2211b05...output omitted...
9.86 MB / 9.86 MB [=====] 0s
Copying blob sha256:aed1801...output omitted...
15.78 MB / 15.78 MB [=====] 1s
Copying blob sha256:7c472a4...output omitted...
300 B / 300 B [=====] 0s
Copying config sha256:b7cc370...output omitted...
7.18 KB / 7.18 KB [=====] 0s
Writing manifest to image destination
Storing signatures
b51444e3b1d7aaaf94b3a4a54485d76a0a094cbfac89c287d360890a3d2779a5a
```

Este comando inicia el servidor HTTP de Apache en segundo plano y vuelve al prompt de Bash.

2. Pruebe el contenedor **httpd-basic**.

Desde **workstation**, intente acceder a `http://localhost:8080` con cualquier navegador web.

Se muestra un mensaje **Hello from the httpd-parent container!** (Hola desde el contenedor httpd-parent), que es la página **index.html** del contenedor del servidor HTTP Apache que se ejecuta en **workstation**.

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

3. Personalice el contenedor **httpd-basic** para mostrar **Hello World** (Hola, mundo) como el mensaje. El mensaje del contenedor se almacena en el archivo `/var/www/html/index.html`.

3.1. Inicie una sesión de Bash dentro del contenedor.

Ejecute el siguiente comando:

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

3.2. En la sesión de Bash, verifique el archivo **index.html** en el directorio `/var/www/html` con el comando `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. Cambie el archivo **index.html** para contener el texto **Hello World** (Hola, mundo), que reemplaza todo el contenido existente.

Desde la sesión de Bash en el contenedor, ejecute el siguiente comando:

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

3.4. Intente acceder a `http://localhost:8080` nuevamente y verifique que se haya actualizado la página web.

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

Evaluación

Evalúe su trabajo ejecutando el comando `lab container-review grade` en su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el script **lab container-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Podman permite que los usuarios busquen y descarguen imágenes de registros locales o remotos.
- El comando **podman run** crea e inicia un nuevo contenedor a partir de una imagen de contenedor.
- Los contenedores se ejecutan en segundo plano con el indicador **-d**, o interactivamente con el indicador **-it**.
- Algunas imágenes de contenedores requieren variables de entorno configuradas con la opción **-e** del comando **podman run**.
- Red Hat Container Catalog ayuda a buscar, explorar y analizar imágenes de contenedores del repositorio de imágenes de contenedor oficial de Red Hat.

capítulo 3

Administración de contenedores

Meta

Modificar las imágenes de contenedores creadas previamente para crear y administrar servicios en contenedores.

Objetivos

- Administrar el ciclo de vida de un contenedor desde su creación hasta su eliminación.
- Guardar los datos de la aplicación de contenedor con almacenamiento persistente.
- Describir cómo usar el reenvío de puertos para acceder a un contenedor.

Secciones

- Administración del ciclo de vida de los contenedores (y ejercicio guiado)
- Conexión del almacenamiento persistente a los contenedores (y ejercicio guiado)
- Acceso a contenedores (y ejercicio guiado)

Trabajo de laboratorio

- Administración de contenedores

Administración del ciclo de vida de los contenedores

Objetivos

Después de completar esta sección, los estudiantes deberían poder administrar el ciclo de vida de un contenedor desde su creación hasta su eliminación.

Administración del ciclo de vida del contenedor con Podman

En los capítulos anteriores, aprendió a usar Podman para crear un servicio en contenedores. Ahora profundizará en los comandos y las estrategias que puede usar para administrar el ciclo de vida de un contenedor. Podman permite no solo ejecutar contenedores, sino también hacer que se ejecuten en segundo plano, ejecutar nuevos procesos dentro de ellos y proporcionarles recursos como volúmenes de sistema de archivos o una red.

Podman, implementado por el comando **podman**, proporciona un conjunto de subcomandos para crear y administrar contenedores. Los desarrolladores usan esos subcomandos para administrar el ciclo de vida de los contenedores y las imágenes de contenedor. En la siguiente figura, se muestra un resumen de los subcomandos usados más comúnmente que cambian el estado del contenedor y la imagen.

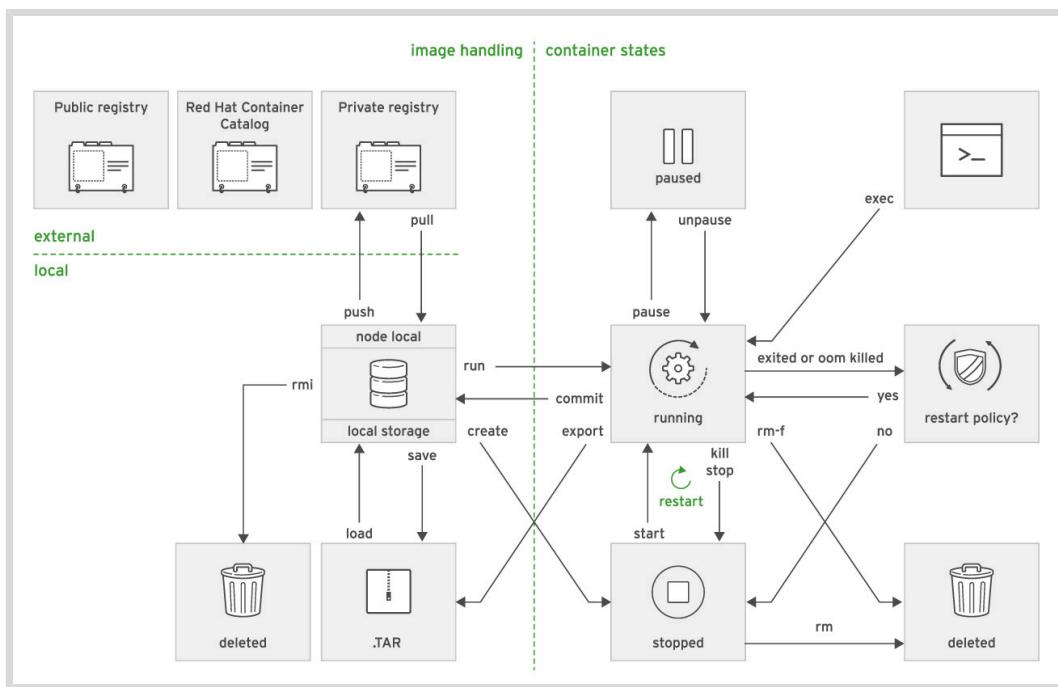


Figura 3.1: Subcomandos de administración de Podman

Podman también proporciona un conjunto de subcomandos útiles para obtener información sobre la ejecución y la detención de contenedores. Puede usar esos subcomandos para extraer información de contenedores e imágenes para fines de depuración, actualización o informes. En la siguiente figura, se muestra un resumen de los subcomandos usados más comúnmente que consultan información en los contenedores y las imágenes.

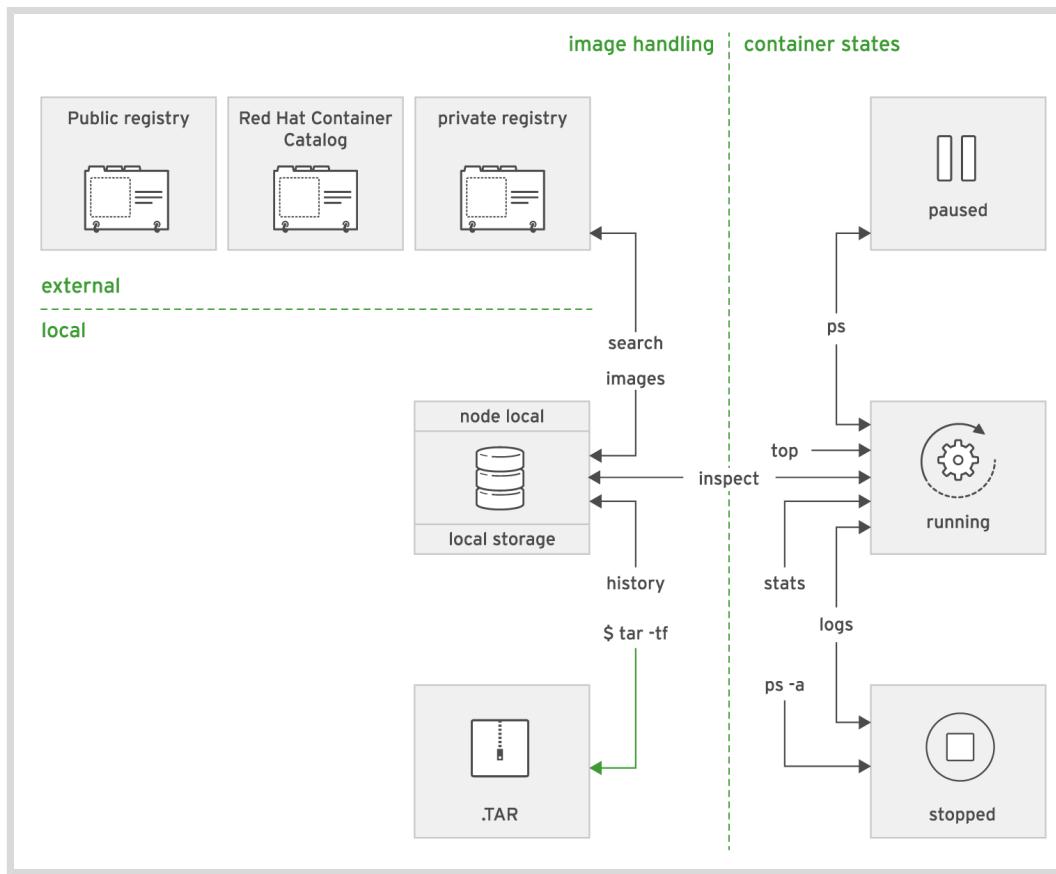


Figura 3.2: Subcomandos de consulta de Podman

Use estas dos figuras como referencia mientras aprende acerca de los subcomandos de Podman en este curso.

Creación de contenedores

El comando **podman run** crea un nuevo contenedor a partir de una imagen e inicia un proceso dentro del nuevo contenedor. Si la imagen de contenedor no está disponible localmente, este comando intenta descargar la imagen usando el repositorio de imágenes configurado:

```
[student@workstation ~]$ sudo podman run rhscl/httpd-24-rhel7
Trying to pull registry...httpd-24-rhel7:latest...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
```

En el ejemplo de salida anterior, el contenedor se inició con un proceso no interactivo (sin la opción **-it**) y se ejecuta en primer plano porque no se inició con la opción **-d**. Por lo tanto, al detener el proceso resultante con **Ctrl+C (SIGINT)**, se detiene tanto el proceso del contenedor como el contenedor en sí.

Podman identifica los contenedores por un ID de contenedor único o un nombre de contenedor. El comando **podman ps** muestra el ID y los nombres de todos los contenedores que se ejecutan activamente:

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID      IMAGE               COMMAND                  ... NAMES
47c9aad6049①    rhscl/httpd-24-rhel7 "httpd -D FOREGROUND" ... focused_fermat②
```

- ① El ID del contenedor es único y se genera automáticamente.
- ② El nombre del contenedor se puede especificar manualmente; de lo contrario, se genera automáticamente. Este nombre debe ser único o el comando **run** falla.

El comando **podman run** genera automáticamente un ID único y aleatorio. También genera un nombre de contenedor aleatorio. Para definir explícitamente el nombre del contenedor, use la opción **--name** al ejecutar un contenedor:

```
[student@workstation ~]$ sudo podman run --name my-httpd-container rhscl/httpd-24-rhel7
...output omitted...AH00094: Command line: 'httpd -D FOREGROUND'
```



nota

El nombre debe ser único. Podman arroja un error si el nombre ya está en uso, incluidos los contenedores detenidos.

Otra función importante es la capacidad de ejecutar el contenedor como proceso de daemon en segundo plano. La opción **-d** es responsable de ejecutarse en modo separado (detached). Al usar esta opción, Podman devuelve el ID del contenedor en la pantalla, lo que le permite continuar ejecutando comandos en el mismo terminal mientras el contenedor se ejecuta en segundo plano:

```
[student@workstation ~]$ sudo podman run --name my-httpd-container -d rhscl/
httpd-24-rhel7
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

La imagen del contenedor especifica el comando que se debe ejecutar para iniciar el proceso del contenedor, conocido como punto de entrada. El comando **podman run** puede anular este punto de entrada al incluir el comando después de la imagen del contenedor:

```
[student@workstation ~]$ sudo podman run rhscl/httpd-24-rhel7 ls /tmp
anaconda-post.log
ks-script-1j4CXN
yum.log
```

El comando especificado debe poder ejecutarse dentro de la imagen de contenedor.



nota

Debido a que un comando especificado aparece en el ejemplo anterior, el contenedor omite el punto de entrada para la imagen **httpd**. Por lo tanto, el servicio **httpd** no se inicia.

Algunos contenedores deben ejecutarse como una shell o un proceso interactivos. Esto incluye a los contenedores que ejecutan procesos que necesitan entradas del usuario (como el ingreso de

comandos) y procesos que generan salidas a través de la salida estándar. En el siguiente ejemplo, se inicia un shell **bash** interactivo en el contenedor **rhscl/httpd-24-rhel7**:

```
[student@workstation ~]$ sudo podman run -it rhscl/httpd-24-rhel7 /bin/bash  
bash-4.2#
```

Las opciones **-t** y **-i** habilitan el redireccionamiento de terminales para los programas interactivos basados en texto. La opción **-t** asigna un **pseudo-tty** (un terminal) y lo conecta a la entrada estándar del contenedor. La opción **-i** mantiene abierta la entrada estándar del contenedor, incluso si estaba desconectada, por lo que el proceso principal puede seguir esperando la entrada.

Ejecución de comandos en un contenedor

Cuando se inicia un contenedor, ejecuta el comando de punto de entrada. Sin embargo, puede ser necesario ejecutar otros comandos para administrar el contenedor que se está ejecutando. Algunos casos de uso típicos se muestran a continuación:

- Ejecución de una shell interactiva en un contenedor ya en ejecución.
- Ejecución de procesos que actualizan o muestran los archivos del contenedor.
- Inicio de nuevos procesos en segundo plano dentro del contenedor.

El comando **podman exec** inicia un proceso adicional dentro de un contenedor que ya está en ejecución:

```
[student@workstation ~]$ sudo podman exec 7ed6e671a600 cat /etc/hostname  
7ed6e671a600
```

El ejemplo anterior usa el ID de contenedor para ejecutar el comando.

Podman recuerda el último contenedor usado en cualquier comando. Los desarrolladores pueden omitir la escritura del ID o el nombre de este contenedor en los comandos posteriores de Podman al reemplazar el ID del contenedor por la opción **-l**:

```
[student@workstation ~]$ sudo podman exec my-httpd-container cat /etc/hostname  
7ed6e671a600  
[student@workstation ~]$ sudo podman exec -l cat /etc/hostname  
7ed6e671a600
```

Administración de contenedores

Crear e iniciar un contenedor es solo el primer paso del ciclo de vida del contenedor. El ciclo de vida de un contenedor también incluye detenerlo, reiniciarlo o, finalmente, eliminarlo. Los usuarios también pueden examinar el estado y los metadatos del contenedor para fines de depuración, actualización o informes.

Podman proporciona los siguientes comandos para administrar contenedores:

- **podman ps**: Este comando muestra los contenedores que se están ejecutando:

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID  IMAGE          COMMAND      CREATED     STATUS      PORTS      NAMES
77d4b7b8ed1f①  rhscl/httpd-24-rhel7②  "httpd..."③  ...ago④  Up...⑤  80/tcp⑥  my-
htt...⑦
```

- ① Cada contenedor, al crearse, obtiene un **ID de contenedor**, que es un número hexadecimal. Este ID parece un identificador de imagen, pero no lo es.
- ② La imagen de contenedor que se usó para iniciar el contenedor.
- ③ El comando que se ejecutó cuando se inició el contenedor.
- ④ La fecha y la hora en que se inició.
- ⑤ Tiempo de actividad total del contenedor, si aún se está ejecutando, o tiempo desde que se finalizó.
- ⑥ Los puertos que expuso el contenedor o los reenvíos de puertos, si se configuraron.
- ⑦ Nombre del contenedor.

Podman no descarta los contenedores detenidos de inmediato. Podman conserva sus sistemas de archivos locales y otros estados para facilitar el análisis *post mortem*. La opción **-a** enumera todos los contenedores, incluidos los detenidos:

```
[student@workstation ~]$ sudo podman ps -a
CONTAINER ID  IMAGE          COMMAND      CREATED     STATUS      PORTS      NAMES
4829d82fbfff  rhscl/httpd-24-rhel7  "httpd..."  ...ago     Exited (0)...  my-
httpd...
```



nota

Al crear contenedores, Podman se cancela si el nombre del contenedor ya está en uso, incluso si el contenedor está en estado “stopped” (detenido). Esta opción puede ayudar a evitar los nombres de contenedores duplicados.

- **podman inspect**: Este comando muestra los metadatos acerca de un contenedor en ejecución o detenido. El comando produce una salida **JSON**:

```
[student@workstation ~]$ sudo podman inspect my-httpd-container
[
{
  "Id": "980e45...76c8be",
  ...output omitted...
  "NetworkSettings": {
    "Bridge": "",
    "EndpointID": "483fc9...5d801a",
    "Gateway": "172.17.42.1",
    "GlobalIPv6Address": "",
    "GlobalIPv6PrefixLen": 0,
    "HairpinMode": false,
    "IPAddress": "172.17.0.9",
    ...output omitted...
}
```

Este comando permite formatear la cadena de salida con la plantilla Go determinada con la opción **-f**. Por ejemplo, para recuperar solo la dirección IP, use el siguiente comando:

```
[student@workstation ~]$ sudo podman inspect \
> -f '{{ .NetworkSettings.IPAddress }}' my-httdp-container
172.17.0.9
```

- **podman stop:** Este comando detiene correctamente un contenedor que se está ejecutando:

```
[student@workstation ~]$ sudo podman stop my-httdp-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

El uso de **podman stop** es más fácil que buscar el proceso de inicio del contenedor en el sistema operativo del host y finalizarlo.

- **podman kill:** Este comando envía señales de Unix al proceso principal del contenedor. Si no se especifica ninguna señal, envía la señal **SIGKILL**, que finaliza el proceso principal y el contenedor.

```
[student@workstation ~]$ sudo podman kill my-httdp-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

Puede especificar la señal con la opción **-s**:

```
[student@workstation ~]$ sudo podman kill -s SIGKILL my-httdp-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

Cualquier señal de Unix puede enviarse al proceso principal. Podman acepta el nombre y el número de la señal. En la siguiente tabla, se muestran varias señales útiles:

Señal	Valor	Acción predeterminada	Comentario
SIGHUP	1	Term	Se detectó un bloqueo en el terminal de control o la desactivación del proceso de control
SIGINT	2	Term	Interrumpir desde el teclado
SIGQUIT	3	Core	Salir desde el teclado
SIGILL	4	Core	Instrucción ilegal
SIGABRT	6	Core	Abortar señal de abortar (3)
SIGFPE	8	Core	Excepción de punto flotante
SIGKILL	9	Term	Señal de finalización inmediata
SIGSEGV	11	Core	Referencia de memoria no válida
SIGPIPE	13	Term	Canalización rota: escritura a canalización sin lectores
SIGALRM	14	Term	Señal del temporizador de alarma (2)

Señal	Valor	Acción predeterminada	Comentario
SIGTERM	15	Term	Señal de terminación
SIGUSR1	30,10,16	Term	Señal definida por el usuario 1
SIGUSR2	31,12,17	Term	Señal definida por el usuario 2
SIGCHLD	20,17,18	Ign	Proceso secundario detenido o terminado
SIGCONT	19,18,25	Cont	Continuar si se había detenido
SIGSTOP	17,19,23	Stop	Detener el proceso
SIGTSTP	18,20,24	Stop	Deja de escribir en tty
SIGTTIN	21,21,26	Stop	Entrada de tty para proceso en segundo plano
SIGTTOU	22,22,27	Stop	Salida de tty para proceso en segundo plano



nota

Term

Finaliza el proceso.

Core

Finaliza el proceso y genera un volcado core.

Ign

La señal se ignora.

Stop

Detiene el proceso.

- **podman restart**: Este comando reinicia un contenedor detenido:

```
[student@workstation ~]$ sudo podman restart my-httdp-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

El comando **podman restart** crea un nuevo contenedor con el mismo ID de contenedor y reutiliza tanto el sistema de archivos como el estado del contenedor detenido.

- **podman rm**: Este comando elimina un contenedor; además, descarta su estado y sistema de archivos:

```
[student@workstation ~]$ sudo podman rm my-httdp-container
77d4b7b8ed1fd57449163bcb0b78d205e70d2314273263ab941c0c371ad56412
```

La opción **-f** del subcomando **rm** le indica a Podman que elimine el contenedor aunque no esté detenido. Esta opción finaliza el contenedor a la fuerza y, luego, lo elimina. La opción **-f** es equivalente a los comandos **podman kill** y **podman rm** juntos.

Puede eliminar todos los contenedores al mismo tiempo. Muchos subcomandos **podman** aceptan la opción **-a**. Esta opción indica el uso del subcomando en todos los contenedores o imágenes disponibles. En el siguiente ejemplo, se eliminan todos los contenedores:

```
[student@workstation ~]$ sudo podman rm -a  
5fd8e98ec7eab567eabe84943fe82e99fdfc91d12c65d99ec760d5a55b8470d6  
716fd687f65b0957edac73b84b3253760e915166d3bc620c4aec8e5f4eadfe8e  
86162c906b44f4cb63ba2e3386554030dc6abedbce9e9fcad60aa9f8b2d5d4
```

Antes de eliminar todos los contenedores, todos los contenedores que se estén ejecutando deben estar en estado "stopped" (detenido). Puede usar el siguiente comando para detener todos los contenedores:

```
[student@workstation ~]$ sudo podman stop -a  
5fd8e98ec7eab567eabe84943fe82e99fdfc91d12c65d99ec760d5a55b8470d6  
716fd687f65b0957edac73b84b3253760e915166d3bc620c4aec8e5f4eadfe8e  
86162c906b44f4cb63ba2e3386554030dc6abedbce9e9fcad60aa9f8b2d5d4
```



nota

Los subcomandos **inspect**, **stop**, **kill**, **restart** y **rm** pueden usar el ID del contenedor en lugar del nombre del contenedor.



Referencias

Página del manual de Unix Posix Signals

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

► Ejercicio Guiado

Administración de un contenedor MySQL

En este ejercicio, creará y administrará un contenedor de bases de datos MySQL®.

Resultados

Debería poder crear y administrar un contenedor de bases de datos MySQL.

Andes De Comenzar

Verifique que **workstation** tenga el comando **podman** disponible y esté bien configurado ejecutando el siguiente comando desde una ventana de terminal:

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

- 1. Abra una ventana de terminal desde la máquina virtual **workstation** (**Applications** [Aplicaciones] → **Utilities** [Utilidades] → **Terminal**) y ejecute el siguiente comando:

```
[student@workstation ~]$ sudo podman run --name mysql-db rhscl/mysql-57-rhel7
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: '^[a-zA-Z0-9_]+$')
  MYSQL_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=<>, .?;:|]+$')
  MYSQL_DATABASE (regex: '^[a-zA-Z0-9_]+$')
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
```

Este comando descarga la imagen de contenedor de bases de datos MySQL e intenta iniciarla, pero no inicia. Esto se debe a que la imagen requiere varias variables de entorno.



nota

Si intenta ejecutar el contenedor como daemon (-d), no se mostrará el mensaje de error acerca de las variables requeridas. Sin embargo, este mensaje se incluye como parte de los registros del contenedor, que pueden verse con el siguiente comando:

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

- 2. Inicie el contenedor nuevamente, y proporcione las variables requeridas: Asígnele el nombre **mysql**. Especifique cada variable con el parámetro **-e**.

**nota**

Asegúrese de iniciar el nuevo contenedor con el nombre correcto.

```
[student@workstation ~]$ sudo podman run --name mysql \
> -d -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> rhscl/mysql-57-rhel7
```

La salida es el ID del contenedor para el contenedor **mysql**. A continuación se muestra un ejemplo de la salida.

```
a49dba9ff17f2b5876001725b581fdd331c9ab8b9eda21cc2a2899c23f078509
```

- 3. Verifique que el contenedor se haya iniciado correctamente. Ejecute el siguiente comando:

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID ...output omitted... STATUS PORTS NAMES
a49dba9ff17f ...output omitted... Up About a minute ago mysql
```

El ID de contenedor que se muestra es una abreviatura del ID de contenedor que se visualiza en el comando anterior.

- 4. Inspeccione los metadatos del contenedor para obtener la dirección IP desde la base de datos MySQL:

```
[student@workstation ~]$ sudo podman inspect \
> -f '{{ .NetworkSettings.IPAddress }}' mysql
10.88.0.6
```

La dirección IP de su contenedor puede diferir de la que se muestra arriba (**10.88.0.6**).

**nota**

Puede obtener otra información importante con el comando **podman inspect**. Por ejemplo, si olvida la contraseña de root, está disponible en la sección **Env**.

▶ 5. Cree la tabla **Projects** (Proyectos):

Está conectado con la base de datos **items**. Cree una tabla nueva con uno de los siguientes:

- Conéctese a la base de datos MySQL y escriba el comando **CREATE TABLE**.
- 1. Conéctese con la base de datos MySQL desde el host. Cambie la dirección IP en el siguiente comando para que coincida con la dirección IP de su contenedor **mysql**:

```
[student@workstation ~]$ mysql -uuser1 -h 10.88.0.6 -p items
Enter password:
```

Use **mypa55** como la contraseña.

```
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.7.16 MySQL Community Server (GPL)

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [items]>
```

2. Escriba o copie los siguientes comandos SQL:

```
MySQL [items]> CREATE TABLE Projects (id int(11) NOT NULL,
-> name varchar(255) DEFAULT NULL, code varchar(255) DEFAULT NULL,
-> PRIMARY KEY (id));
Query OK, 0 rows affected (0.05 sec)
```

3. Inserte una fila en la tabla:

```
MySQL [items]> insert into Projects (id, name, code) values (1,'DevOps','D0180');
Query OK, 1 row affected (0.09 sec)
```

4. Salga del prompt de MySQL:

```
MySQL [items]> exit
```

- De manera alternativa, puede crear la base de datos e insertar la fila con el archivo proporcionado:

```
[student@workstation ~]$ mysql -uuser1 -h 10.88.0.6 \
> -pmypa55 items < D0180/labs/manage-lifecycle/db.sql
```

- 6. Cree otro contenedor con la misma imagen de contenedor del contenedor anterior mediante la ejecución de la shell de **/bin/bash**:

```
[student@workstation ~]$ sudo podman run --name mysql-2 \
> -it rhsc1/mysql-57-rhel7 /bin/bash
bash-4.2$
```

- 7. Intente conectarse con la base de datos MySQL del contenedor nuevo:

```
bash-4.2$ mysql -uroot
```

Se muestra el siguiente error:

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

El motivo de este error es que el servidor de bases de datos MySQL no se está ejecutando porque, cuando creamos el contenedor nuevo, modificamos el punto de entrada responsable de iniciar la base de datos a **/bin/bash**.

- 8. Salga de la shell **bash**:

```
bash-4.2$ exit
```

- 9. Verifique que el contenedor **mysql-2** no esté ejecutándose:

```
[student@workstation ~]$ sudo podman ps -a \
> --format="table {{.ID}} {{.Names}} {{.Status}}"
CONTAINER ID NAMES STATUS
2871e392af02 mysql-2 Exited (1) 19 seconds ago
a49dba9ff17f mysql Up 10 minutes ago
c053c7e09c21 mysql-db Exited (1) 44 minutes ago
```

- 10. Ejecute comandos en el contenedor separado. Use **mypa55** como la contraseña:

```
[student@workstation ~]$ sudo podman exec mysql /bin/bash \
> -c 'mysql -uuser1 -p -e "select * from items.Projects;"'
Enter password: mypa55
id name code
1 DevOps D0180
```

El comando anterior ejecuta un intérprete de **bash** en el contenedor **mysql**. A continuación, el comando le indica a bash que ejecute el intérprete de **mysql**, que recibe la consulta de SQL para obtener datos de la base de datos.

Finalizar

En **workstation**, ejecute el script **lab manage-lifecycle finish** para terminar este ejercicio.

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

Esto concluye el ejercicio.

Conexión de almacenamiento persistente a los contenedores

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Guardar datos de la aplicación durante los reinicios de los contenedores mediante el uso de almacenamiento persistente.
- Configurar directorios de host para usar como volúmenes del contenedor.
- Montar un volumen dentro del contenedor.

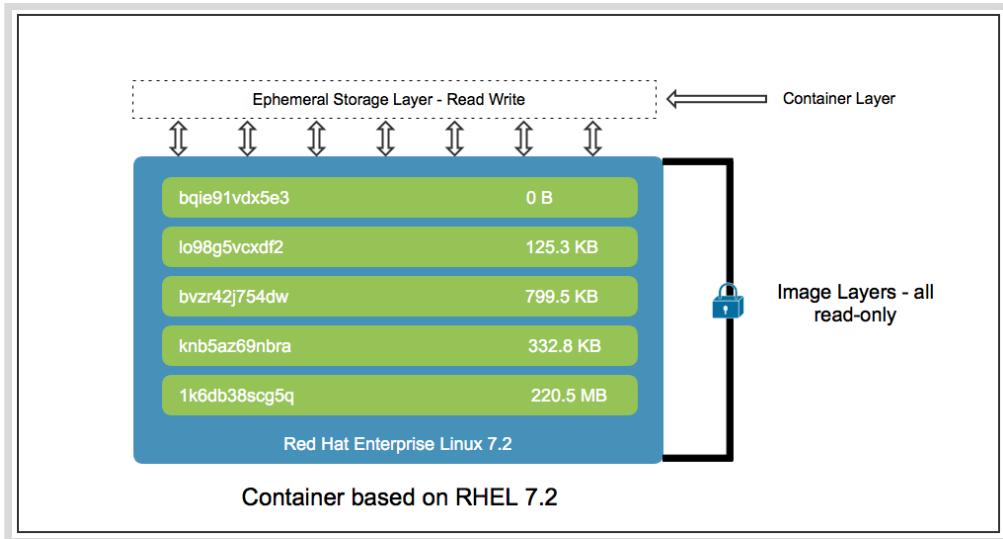
Preparación de las ubicaciones de almacenamiento persistente

Se dice que el almacenamiento del contenedor es *efímero*, lo que significa que su contenido no se conserva después de que se elimina el contenedor. Las aplicaciones en contenedores funcionan sobre el supuesto de que se inician siempre con el almacenamiento vacío, y esto hace que la creación y la destrucción de contenedores sean operaciones relativamente económicas.

Anteriormente en este curso, las imágenes de contenedores se caracterizaban como *inmutables* y *con capas*, lo que significa que nunca se modifican, sino que están compuestas por capas que agregan o reemplazan el contenido de capas inferiores.

Un contenedor en ejecución obtiene una nueva capa sobre su imagen de contenedor base, y esta capa es el *almacenamiento del contenedor*. Al principio, esta capa es el almacenamiento de solo lectura-escritura disponible para el contenedor y se usa para crear archivos de trabajo, archivos temporales y archivos de registro. Estos archivos se consideran volátiles. Una aplicación no deja de funcionar si estos se pierden. La capa del almacenamiento del contenedor es exclusiva del contenedor en ejecución, de modo que si se crea otro contenedor a partir de la misma imagen base, obtiene otra capa de lectura-escritura. Esto garantiza que los recursos de cada contenedor estén aislados de otros contenedores similares.

El almacenamiento efímero del contenedor no es suficiente para las aplicaciones que necesitan conservar los datos durante los reinicios, como las bases de datos. Para soportar estas aplicaciones, el administrador debe proporcionar un contenedor con almacenamiento persistente.

**Figura 3.3: Capas del contenedor**

Las aplicaciones en contenedores no deberían intentar usar el almacenamiento del contenedor para almacenar datos persistentes, ya que no pueden controlar el tiempo durante el cual se conservará su contenido. Aun si fuera posible mantener indefinidamente el almacenamiento del contenedor, el sistema de archivos por capas no funciona bien para cargas de trabajo con uso intensivo de E/S y no sería adecuado para la mayoría de las aplicaciones que requieren almacenamiento persistente.

Recuperación del almacenamiento

Podman mantiene el almacenamiento del contenedor detenido anterior disponible por un tiempo para usarlo para las operaciones de solución de problemas, como la revisión de los registros de un contenedor fallido en busca de mensajes de error.

Si el administrador necesita recuperar el almacenamiento de contenedores anteriores, puede eliminar el contenedor con **podman rm container_id**. Este comando también elimina el almacenamiento del contenedor. Los identificadores de contenedores detenidos se pueden encontrar con el comando **podman ps -a**.

Preparación del directorio del host

Podman puede montar directorios de host dentro de un contenedor en ejecución. La aplicación en contenedores ve los directorios del host como parte del almacenamiento del contenedor, al igual que las aplicaciones normales ven el volumen de red remoto como si fuera parte del sistema de archivos del host. Sin embargo, el contenido de estos directorios del host no se recupera después de que se detenga el contenedor y podrá montarse en nuevos contenedores siempre que se necesite.

Por ejemplo, un contenedor de bases de datos puede usar un directorio del host para almacenar archivos de la base de datos. Si este contenedor de bases de datos falla, Podman puede crear un nuevo contenedor con el mismo directorio del host y mantener los datos de la base de datos disponibles para las aplicaciones de clientes. Para el contenedor de la base de datos, no importa dónde se almacene este directorio del host desde el punto de vista del host; podría ser cualquier cosa, desde una partición de un disco duro local hasta un sistema de archivos remoto conectado en red.

Un contenedor se ejecuta como un proceso del sistema operativo del host, con un ID de grupo y de usuario del sistema operativo del host, de modo que el directorio del host debe configurarse

con propiedad y permisos que permitan el acceso al contenedor. En RHEL, el directorio del host también debe configurarse con el contexto de SELinux correcto, que es **container_file_t**. Podman usa el contexto de SELinux **container_file_t** para restringir a qué archivos del sistema host puede acceder el contenedor. Esto evita la fuga de información entre el sistema host y las aplicaciones que se ejecutan dentro de contenedores.

A continuación se describe una manera de configurar el directorio del host:

1. Cree un directorio con el propietario y grupo **root**:

```
[student@workstation ~]$ sudo mkdir /var/dbfiles
```

2. El usuario que ejecuta procesos en el contenedor debe ser capaz de escribir archivos en el directorio. Si la máquina del host no tiene el mismo usuario exacto definido, el permiso debe definirse con el ID de usuario numérico (UID) desde el contenedor. En el caso del servicio MySQL provisto por Red Hat, el UID es 27:

```
[student@workstation ~]$ sudo chown -R 27:27 /var/dbfiles
```

3. Aplique el contexto de **container_file_t** al directorio (y a todos los subdirectorios) para permitir que los contenedores accedan a todo su contenido.

```
[student@workstation ~]$ sudo semanage fcontext -a -t container_file_t '/var/dbfiles(/.*)?'
```

4. Aplique la política de contenedores de SELinux que configuró en el primer paso al directorio recién creado:

```
[student@workstation ~]$ sudo restorecon -Rv /var/dbfiles
```

El directorio del host debe configurarse *antes* de iniciar el contenedor que lo usa.

Montaje de un volumen

Después de crear y configurar el directorio del host, el siguiente paso es montar este directorio en un contenedor. Para montar tipo enlace un directorio del host en un contenedor, agregue la opción **-v** al comando **podman run** y especifique la ruta del directorio del host y la ruta de almacenamiento del contenedor, separadas por dos puntos (:).

Por ejemplo, para usar el directorio del host **/var/dbfiles** para los archivos de la base de datos del servidor de MySQL, que se espera que estén en **/var/lib/mysql** dentro de la imagen de contenedor de MySQL nombrada **mysql**, use el siguiente comando:

```
[student@workstation ~]$ sudo podman run -v /var/dbfiles:/var/lib/mysql rhmap47/mysql
```

En el comando anterior, si **/var/lib/mysql** ya existe dentro de la imagen de contenedor **mysql**, el montaje de **/var/dbfiles** se superpone pero no elimina el contenido de la imagen de contenedor. Si se elimina el montaje, el contenido original está accesible nuevamente.

► Ejercicio Guiado

Persistencia de una base de datos MySQL

En este ejercicio, creará un contenedor que almacene los datos de la base de datos MySQL en un directorio del host.

Resultados

Deberá ser capaz de implementar un contenedor con una base de datos persistente.

Andes De Comenzar

En **workstation**, no debería haber ninguna imagen de contenedor ejecutándose. Ejecute el comando siguiente en **workstation**:

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

- ▶ 1. Abra una ventana de terminal en **workstation** (**Applications [Aplicaciones]** → **System Tools [Herramientas del sistema]** → **Terminal**).
- ▶ 2. Cree el directorio **/var/local/mysql** con el contexto y los permisos correctos de SELinux.
 - 2.1. Cree el directorio **/var/local/mysql**.

```
[student@workstation ~]$ sudo mkdir -pv /var/local/mysql
mkdir: created directory '/var/local/mysql'
```

- 2.2. Agregue el contexto de SELinux adecuado para el directorio **/var/local/mysql** y su contenido.

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

- 2.3. Aplique la política de SELinux al directorio recién creado.

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

- 2.4. Verifique que el tipo de contexto de SELinux para el directorio **/var/local/mysql** sea **container_file_t**.

```
[student@workstation ~]$ ls -dz /var/local/mysql
drwxr-xr-x. root root unconfined_u:object_r:container_file_t:s0 /var/local/mysql
```

- 2.5. Cambie el propietario del directorio **/var/local/mysql** al usuario **mysql** y al grupo **mysql**.

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

**nota**

El usuario que ejecuta procesos en el contenedor debe ser capaz de escribir archivos en el directorio. Si la máquina del host no tiene el mismo usuario exacto definido, el permiso debe definirse con el ID de usuario numérico (UID) desde el contenedor. En el caso del servicio MySQL provisto por Red Hat, el UID es 27.

- 3. Cree una instancia del contenedor MySQL con almacenamiento persistente.

- 3.1. Extraiga la imagen de contenedor de MySQL desde el registro interno:

```
[student@workstation ~]$ sudo podman pull rhscl/mysql-57-rhel7
Trying to pull ...output omitted...rhscl/mysql-57-rhel7...output omitted...
...output omitted...
Writing manifest to image destination
Storing signatures
4ae3a3f4f409a8912cab9fbf71d3564d011ed2e68f926d50f88f2a3a72c809c5
```

- 3.2. Cree un nuevo contenedor y especifique el punto de montaje para almacenar los datos de la base de datos:

```
[student@workstation ~]$ sudo podman run --name persist-db \
> -d -v /var/local/mysql:/var/lib/mysql/data \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> rhscl/mysql-57-rhel7
```

Este comando monta el directorio **/var/local/mysql** del host en el directorio **/var/lib/mysql/data** del contenedor. **/var/lib/mysql/data** es el directorio donde la base de datos MySQL almacena los datos.

- 3.3. Verifique que el contenedor se haya iniciado correctamente. Ejecute el siguiente comando:

```
[student@workstation ~]$ sudo podman ps \
> --format="table {{.ID}} {{.Names}} {{.Status}}"
CONTAINER ID NAMES STATUS
ce637c4da16 persist-db Up 3 minutes ago
```

- 4. Verifique que el directorio **/var/local/mysql** contenga un directorio **items**:

```
[student@workstation ~]$ ls -l /var/local/mysql
total 41032
-rw-r----- 1 27 27 56 Jun 13 07:50 auto.cnf
-rw----- 1 27 27 1676 Jun 13 07:50 ca-key.pem
-rw-r--r-- 1 27 27 1075 Jun 13 07:50 ca.pem
-rw-r--r-- 1 27 27 1079 Jun 13 07:50 client-cert.pem
-rw----- 1 27 27 1680 Jun 13 07:50 client-key.pem
```

```
-rw-r----. 1 27 27 2 Jun 13 07:51 e89494e64d5b.pid
-rw-r----. 1 27 27 349 Jun 13 07:51 ib_buffer_pool
-rw-r----. 1 27 27 12582912 Jun 13 07:51 ibdata1
-rw-r----. 1 27 27 8388608 Jun 13 07:51 ib_logfile0
-rw-r----. 1 27 27 8388608 Jun 13 07:50 ib_logfile1
-rw-r----. 1 27 27 12582912 Jun 13 07:51 ibtmp1
drwxr-x---. 2 27 27 20 Jun 13 07:50 items
drwxr-x---. 2 27 27 4096 Jun 13 07:50 mysql
drwxr-x---. 2 27 27 8192 Jun 13 07:50 performance_schema
-rw-----. 1 27 27 1680 Jun 13 07:50 private_key.pem
-rw-r--r--. 1 27 27 452 Jun 13 07:50 public_key.pem
-rw-r--r--. 1 27 27 1079 Jun 13 07:50 server-cert.pem
-rw-----. 1 27 27 1676 Jun 13 07:50 server-key.pem
drwxr-x---. 2 27 27 8192 Jun 13 07:50 sys
```

Este directorio almacena datos relacionados con la base de datos **items** que creó este contenedor. Si este directorio no está disponible, el punto de montaje no se definió correctamente en la creación del contenedor.

Finalizar

En **workstation**, ejecute el script **lab manage-storage finish** para terminar este trabajo de laboratorio.

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

Esto concluye el ejercicio.

Acceso a los contenedores

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Describir los aspectos básicos de las redes con contenedores.
- Conectarse con servicios dentro de un contenedor de forma remota.

Introducción a las redes con contenedores

Cloud Native Computing Foundation (CNCF) patrocina el proyecto de código abierto de la *Interfaz de redes de contenedores (CNI)*. El proyecto CNI tiene como objetivo estandarizar la interfaz de red para contenedores en entornos nativos de nube, como Kubernetes y Red Hat OpenShift Container Platform.

Podman usa el proyecto CNI para implementar una *red definida por software (SDN)* para contenedores en cada host. Podman adjunta cada contenedor a un puente virtual y le asigna una dirección IP privada a cada contenedor. El archivo de configuración que especifica la configuración de CNI para Podman es **/etc/cni/net.d/87-podman-bridge.conf**.

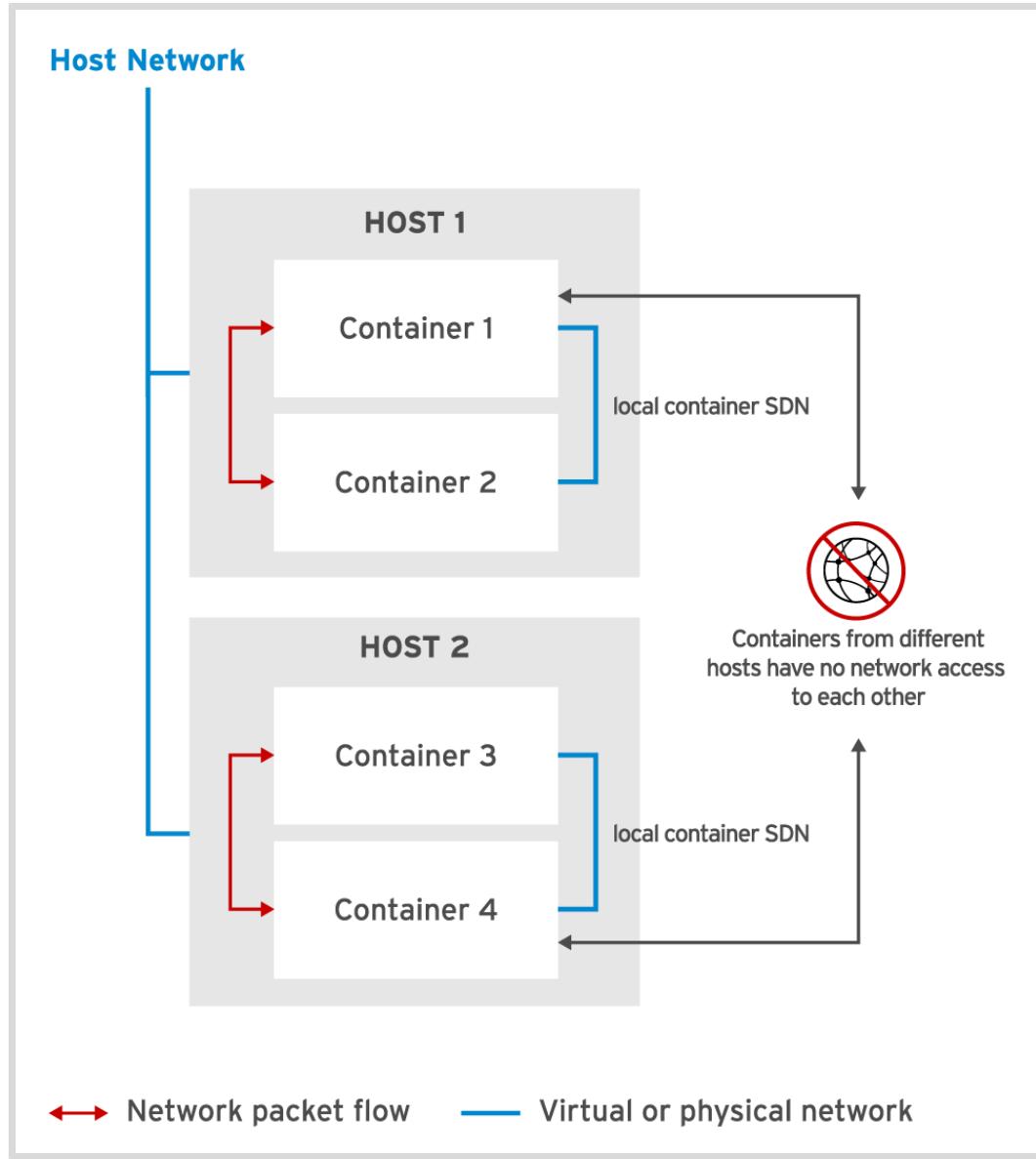


Figura 3.4: Redes básicas de contenedores de Linux

Cuando Podman crea contenedores en el mismo host, asigna a cada contenedor una dirección IP única y los conecta a todos a la misma red definida por software. Estos contenedores pueden comunicarse libremente entre sí por la dirección IP.

Los contenedores creados con Podman que se ejecutan en diferentes hosts pertenecen a diferentes redes definidas por software. Cada SDN está aislada, lo que impide que un contenedor de una red se comunique con un contenedor de una red diferente. Debido al aislamiento de la red, un contenedor en una SDN puede tener la misma dirección IP que un contenedor en una SDN diferente.

También es importante tener en cuenta que, de forma predeterminada, todas las redes de contenedores están ocultas de la red del host. Es decir, los contenedores normalmente pueden acceder a la red del host, pero sin una configuración explícita, no tienen acceso a la red de contenedores.

Administración de puertos de red

Acceder a un contenedor desde la red del host puede ser un desafío. A un contenedor se le asigna una dirección IP de un pool (grupo) de direcciones disponibles. Cuando se destruye un contenedor, la dirección del contenedor se devuelve al pool (grupo) de direcciones disponibles. Otro problema es que solo se puede acceder a la red definida por software del contenedor desde el host del contenedor.

Para resolver estos problemas, defina las reglas de reenvío de puertos para permitir el acceso externo a un servicio de contenedor. Use la opción **-p [<IP address>:]<host port>:<container port>** con el comando **podman run** para crear un contenedor accesible externamente. Considere el siguiente ejemplo:

```
[student@workstation ~]$ sudo podman run -d --name apache1 -p 8080:80 rhscl/
httpd-24-rhel7:2.4
```

El valor **8080:80** especifica que todas las solicitudes al puerto 8080 en el host se reenvían al puerto 80 dentro del contenedor.

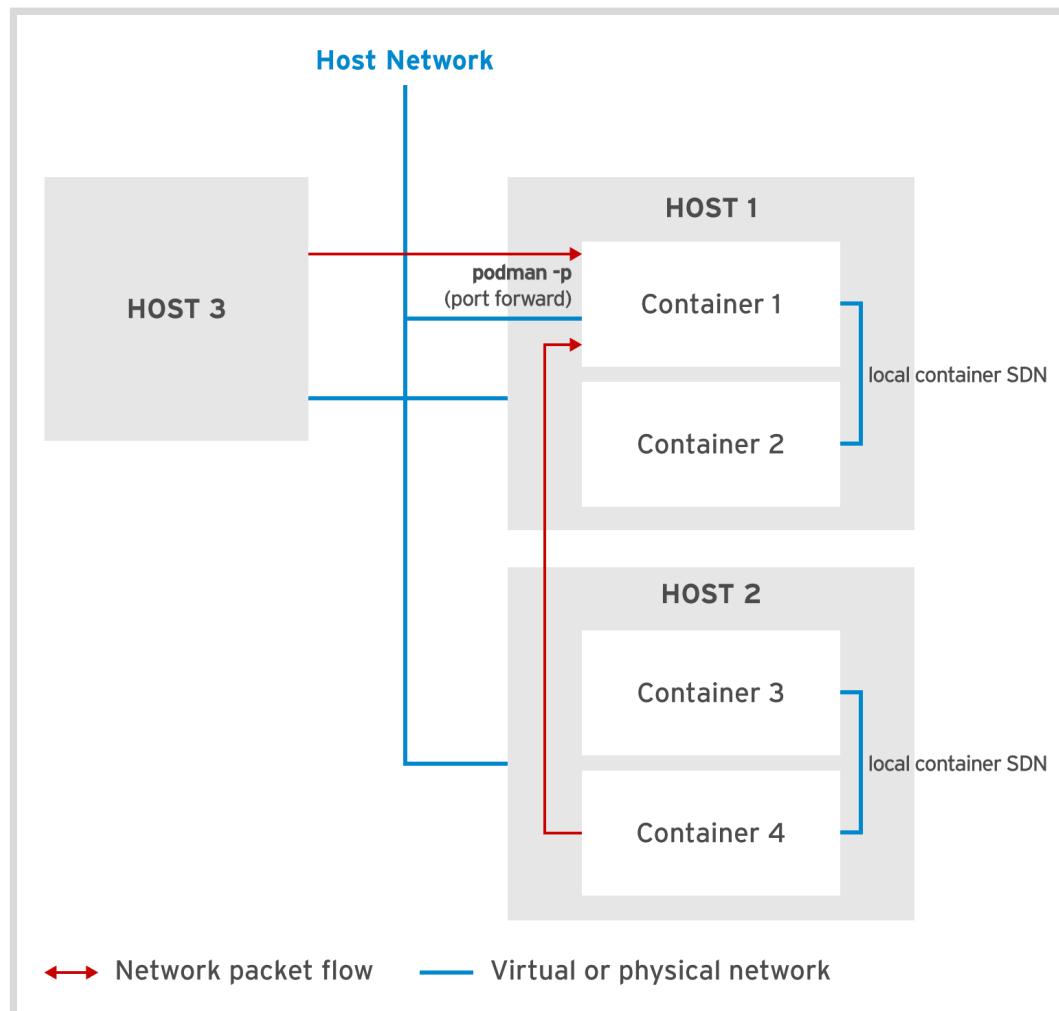


Figura 3.5: Permiso de acceso externo a contenedores de Linux

También puede usar la opción **-p** para reenviar las solicitudes a un contenedor solo si dichas solicitudes provienen de una dirección IP específica:

```
[student@workstation ~]$ sudo podman run -d --name apache2 \
> -p 127.0.0.1:8081:80 rhscl/httpd-24-rhel7:2.4
```

El ejemplo anterior limita el acceso externo al contenedor **apache2** a las solicitudes de **localhost** al puerto de host 8081. Estas solicitudes se reenvían al puerto 80 en el contenedor **apache2**.

Si no se especifica un puerto para el puerto de host, Podman asigna un puerto de host disponible aleatorio para el contenedor:

```
[student@workstation ~]$ sudo podman run -d --name apache3 -p 127.0.0.1::80 rhscl/
httpd-24-rhel7:2.4
```

Para ver el puerto asignado por Podman, use el comando **podman port <container name>**:

```
[student@workstation ~]$ sudo podman port apache3
80/tcp -> 127.0.0.1:35134
[student@workstation ~]$ curl 127.0.0.1:35134
<html><body><h1>It works!</h1></body></html>
```

Si hay un solo puerto de contenedor especificado con la opción **-p**, se asigna al contenedor un puerto de host disponible aleatorio. Las solicitudes a este puerto de host asignado desde cualquier dirección IP se reenvían al puerto de contenedor.

```
[student@workstation ~]$ sudo podman run -d --name apache4 -p 80 rhscl/httpd-24-
rhel7:2.4
[student@workstation ~]$ sudo podman port apache4
80/tcp -> 0.0.0.0:37068
```

En el ejemplo anterior, todas las solicitudes enrutables al puerto de host **37068** se reenvían al puerto 80 en el contenedor.



Referencias

Interfaz de red de contenedor: redes para contenedores Linux

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

Cloud Native Computing Foundation

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

► Ejercicio Guiado

Cargar la base de datos

En este ejercicio, creará un contenedor de bases de datos MySQL. Reenviará puertos desde el contenedor al host para cargar la base de datos con un script SQL.

Resultados

Debería poder implementar un contenedor de bases de datos y cargar un script SQL.

Andes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

Esto asegura que el directorio **/var/local/mysql** existe y está configurado con los permisos correctos para habilitar el almacenamiento persistente para el contenedor MySQL.

- ▶ 1. Cree una instancia de contenedor MySQL con almacenamiento persistente y reenvío de puertos:

```
[student@workstation ~]$ sudo podman run --name mysql-db-port \
> -d -v /var/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 \
> rhscl/mysql-57-rhel7
Trying to pull ...output omitted...
Copying blob sha256:e373541...output omitted...
 69.66 MB / 69.66 MB [=====] 8s
Copying blob sha256:c5d2e94...output omitted...
 1.20 KB / 1.20 KB [=====] 0s
Copying blob sha256:b3949ae...output omitted...
 62.03 MB / 62.03 MB [=====] 8s
Writing manifest to image destination
Storing signatures
9941...7d89
```

La última línea de su salida será diferente de la que se muestra arriba, así como el tiempo necesario para descargar cada capa de imagen.

La opción **-p** configura el reenvío de puertos. En este caso, todas las conexiones de la dirección IP del host al puerto 13306 se reenvían al puerto de contenedor 3306.



nota

El directorio **/var/local/mysql** se creó y se configuró mediante el script de inicio para que tenga los permisos requeridos por la base de datos en contenedores.

- 2. Ejecute el siguiente comando:

```
[student@workstation ~]$ sudo podman ps  
CONTAINER ID ...output omitted... PORTS NAMES  
9941da2936a5 ...output omitted... 0.0.0.0:13306->3306/tcp mysqldb-port
```

Inspeccione la columna **PORTS** (PUERTOS) para ver la regla de reenvío de puertos.

- 3. Cargue la base de datos:

```
[student@workstation ~]$ mysql -uuser1 -h 127.0.0.1 -pmypa55 \  
> -P13306 items < /home/student/D0180/labs/manage-networking/db.sql
```

Si no hay errores, el comando anterior no devuelve ninguna salida.

- 4. Verifique que la base de datos se haya cargado correctamente con una de las tres alternativas siguientes:

- Con el puerto reenviado desde el host y el cliente de base de datos local:

```
[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 |
+---+-----+---+
```

- Ejecute un terminal interactivo y el cliente de base de datos desde dentro del contenedor:

1. Abra una shell de Bash dentro del contenedor.

```
[student@workstation ~]$ sudo podman exec -it mysql5db-port /bin/bash
bash-4.2$
```

2. Verifique que el comando **mysql** esté instalado en el contenedor:

```
bash-4.2$ which mysql
/opt/rh/rh-mysql57/root/usr/bin/mysql
```

3. Verifique que la base de datos contenga datos:

```
bash-4.2$ mysql -uroot items -e "SELECT * FROM Item"
+---+-----+---+
| id | description | done |
+---+-----+---+
| 1 | Pick up newspaper | 0 |
| 2 | Buy groceries | 1 |
+---+-----+---+
```

4. Salga de la shell de Bash dentro del contenedor:

```
bash-4.2$ exit
[student@workstation ~]$
```

- Inserte el proceso **mysql** dentro del contenedor.

```
[student@workstation ~]$ sudo podman exec -it mysql5db-port \
> /opt/rh/rh-mysql57/root/usr/bin/mysql -uroot items -e "SELECT * FROM Item"
+----+-----+----+
| id | description | done |
+----+-----+----+
| 1 | Pick up newspaper | 0 |
| 2 | Buy groceries | 1 |
+----+-----+----+
```



nota

El comando **mysql** no se encuentra en la variable **PATH** y, por este motivo, debe usar una ruta absoluta.

Finalizar

En **workstation**, ejecute el script **lab manage-networking finish** para terminar este trabajo de laboratorio.

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

Esto concluye el ejercicio.

► Trabajo de laboratorio

Administración de contenedores

Listado de verificación de rendimiento

En este trabajo de laboratorio, implementará un contenedor que guarde los datos de la base de datos MySQL en una carpeta del host, cargue la base de datos y administre el contenedor.

Resultados

Debería poder implementar y administrar una base de datos persistente con un volumen compartido. También debe ser capaz de iniciar una segunda base de datos con el mismo volumen compartido y observar que los datos son coherentes entre los dos contenedores porque están usando el mismo directorio del host para almacenar los datos de MySQL.

Antes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Cree el directorio **/var/local/mysql** con el contexto y los permisos correctos de SELinux.
 - 1.1. Cree el directorio **/var/local/mysql**.
 - 1.2. Agregue el contexto de SELinux adecuado para el directorio **/var/local/mysql** y su contenido. Con el contexto correcto, puede montar este directorio en un contenedor en ejecución.
 - 1.3. Aplique la política de SELinux al directorio recién creado.
 - 1.4. Cambie el propietario del directorio **/var/local/mysql** para que coincida con el usuario **mysql** y el grupo **mysql** para la imagen de contenedor **rhscl/mysql-57-rhel7**:
2. Implemente una instancia del contenedor MySQL con las siguientes características:
 - **Nombre:** **mysql-1**
 - **Ejecutar como daemon:** sí
 - **Volumen:** desde la carpeta del host **/var/local/mysql** hasta la carpeta del contenedor **/var/lib/mysql/data**
 - **Imagen de contenedor:** **rhscl/mysql-57-rhel7**
 - **Reenvío de puertos:** no
 - **Variables de entorno:**
 - **MYSQL_USER:** **user1**

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

3. Cargue la base de datos **items** con el script **/home/student/D0180/labs/manage-review/db.sql**.

3.1. Obtenga la dirección IP del contenedor.

3.2. Cargue la base de datos con los comandos SQL en **/home/student/D0180/labs/manage-review/db.sql**. Use la dirección IP devuelta por el comando anterior como la IP del host del servidor de la base de datos.



nota

Puede importar todos los comandos del archivo anterior con el operador menor que (<) después del comando **mysql**, en lugar de escribirlos. Además, debe agregar el parámetro **-h CONTAINER_IP** al comando **mysql** para conectarse al contenedor correcto.

3.3. Use una declaración SQL **SELECT** para generar todas las filas de la tabla de ítems para verificar que la base de datos de ítems esté cargada.



nota

Puedes agregar el parámetro **-e SQL** al comando **mysql** para ejecutar una instrucción SQL.

4. Detenga el contenedor de manera correcta.



Importante

Este paso es muy importante, ya que se creará un nuevo contenedor que compartirá el mismo volumen para los datos de la base de datos. Tener dos contenedores que usen el mismo volumen puede dañar la base de datos. No reinicie el contenedor **mysql-1**.

5. Cree un nuevo contenedor con las siguientes características:

- **Nombre:** mysql-2
- **Ejecutar como daemon:** sí
- **Volumen:** desde la carpeta del host **/var/local/mysql** hasta la carpeta del contenedor **/var/lib/mysql/data**
- **Imagen de contenedor:** **rhscl/mysql-57-rhel7**
- **Reenvío de puertos:** sí, desde el puerto 13306 del host hasta el puerto 3306 del contenedor
- **Variables de entorno:**
 - **MYSQL_USER:** user1

- **MYSQL_PASSWORD: mypa55**
 - **MYSQL_DATABASE: items**
 - **MYSQL_ROOT_PASSWORD: r00tpa55**
6. Guarde la lista de todos los contenedores (incluidos los detenidos) en el archivo **/tmp/my-containers**.
 7. Acceda a la shell de Bash dentro del contenedor y verifique que la base de datos **items** y la tabla **Item** (ítem) aún estén disponibles. Además, confirme que la tabla contiene datos.
 - 7.1. Acceda a la shell de Bash dentro del contenedor.
 - 7.2. Conéctese al servidor MySQL.
 - 7.3. Enumere todas las bases de datos y confirme que la base de datos **items** está disponible.
 - 7.4. Enumere todas las tablas de la base de datos **items** y verifique que la tabla **Item** esté disponible.
 - 7.5. Vea los datos de la tabla.
 - 7.6. Salga del cliente MySQL y de la shell del contenedor.
 8. Mediante el reenvío de puertos, inserte una nueva fila en la tabla **Item**. La fila debe tener un valor de **description** de **Finished lab** (Trabajo de laboratorio finalizado) y un valor de **done** de **1**.
 - 8.1. Conéctese con la base de datos MySQL.
 - 8.2. Inserte la nueva fila.
 - 8.3. Salga del cliente MySQL.
 9. Dado que ya no se necesita el primer contenedor, elimínelo para liberar recursos.

Evaluación

Evalúe su trabajo ejecutando el comando **lab manage-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el comando **lab manage-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

► Solución

Administración de contenedores

Lista de verificación de rendimiento

En este trabajo de laboratorio, implementará un contenedor que guarde los datos de la base de datos MySQL en una carpeta del host, cargue la base de datos y administre el contenedor.

Resultados

Debería poder implementar y administrar una base de datos persistente con un volumen compartido. También debe ser capaz de iniciar una segunda base de datos con el mismo volumen compartido y observar que los datos son coherentes entre los dos contenedores porque están usando el mismo directorio del host para almacenar los datos de MySQL.

Andes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Cree el directorio **/var/local/mysql** con el contexto y los permisos correctos de SELinux.
 - 1.1. Cree el directorio **/var/local/mysql**.

```
[student@workstation ~]$ sudo mkdir -pv /var/local/mysql
mkdir: created directory '/var/local/mysql'
```

- 1.2. Agregue el contexto de SELinux adecuado para el directorio **/var/local/mysql** y su contenido. Con el contexto correcto, puede montar este directorio en un contenedor en ejecución.

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

- 1.3. Aplique la política de SELinux al directorio recién creado.

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

- 1.4. Cambie el propietario del directorio **/var/local/mysql** para que coincida con el usuario **mysql** y el grupo **mysql** para la imagen de contenedor **rhsc1/mysql-57-rhel7**:

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

2. Implemente una instancia del contenedor MySQL con las siguientes características:

- **Nombre:** mysql-1
- **Ejecutar como daemon:** sí
- **Volumen:** desde la carpeta del host `/var/local/mysql` hasta la carpeta del contenedor `/var/lib/mysql/data`
- **Imagen de contenedor:** rhscl/mysql-57-rhel7
- **Reenvío de puertos:** no
- **Variables de entorno:**
 - `MYSQL_USER`: user1
 - `MYSQL_PASSWORD`: mypa55
 - `MYSQL_DATABASE`: items
 - `MYSQL_ROOT_PASSWORD`: r00tpa55

2.1. Cree e inicie el contenedor.

```
[student@workstation ~]$ sudo podman run --name mysql-1 \
> -d -v /var/local/mysql:/var/lib/mysql/data \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> rhscl/mysql-57-rhel7
Trying to pull ...output omitted...
...output omitted...
Writing manifest to image destination
Storing signatures
6l6azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cdd
```

2.2. Verifique que el contenedor se haya iniciado correctamente.

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID      ...output omitted... NAMES
6l6azfaa55x8      ...output omitted... mysql-1
```

3. Cargue la base de datos **items** con el script `/home/student/D0180/labs/manage-review/db.sql`.

3.1. Obtenga la dirección IP del contenedor.

```
[student@workstation ~]$ sudo podman inspect \
> -f '{{ .NetworkSettings.IPAddress }}' mysql-1
10.88.0.6
```

3.2. Cargue la base de datos con los comandos SQL en `/home/student/D0180/labs/manage-review/db.sql`. Use la dirección IP devuelta por el comando anterior como la IP del host del servidor de la base de datos.

**nota**

Puede importar todos los comandos del archivo anterior con el operador menor que (<) después del comando **mysql**, en lugar de escribirlos. Además, debe agregar el parámetro **-h CONTAINER_IP** al comando **mysql** para conectarse al contenedor correcto.

```
[student@workstation ~]$ mysql -uuser1 -h CONTAINER_IP \
> -pmypa55 items < /home/student/D0180/labs/manage-review/db.sql
```

- 3.3. Use una declaración SQL **SELECT** para generar todas las filas de la tabla de ítems para verificar que la base de datos de ítems esté cargada.

**nota**

Puedes agregar el parámetro **-e SQL** al comando **mysql** para ejecutar una instrucción SQL.

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

4. Detenga el contenedor de manera correcta.

**Importante**

Este paso es muy importante, ya que se creará un nuevo contenedor que compartirá el mismo volumen para los datos de la base de datos. Tener dos contenedores que usen el mismo volumen puede dañar la base de datos. No reinicie el contenedor **mysql-1**.

Use el comando siguiente para detener el contenedor:

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

5. Cree un nuevo contenedor con las siguientes características:

- **Nombre:** **mysql-2**
- **Ejecutar como daemon:** sí
- **Volumen:** desde la carpeta del host **/var/local/mysql** hasta la carpeta del contenedor **/var/lib/mysql/data**

- **Imagen de contenedor:** `rhscl/mysql-57-rhel7`
- **Reenvío de puertos:** sí, desde el puerto 13306 del host hasta el puerto 3306 del contenedor
- **Variables de entorno:**
 - `MYSQL_USER: user1`
 - `MYSQL_PASSWORD: mypa55`
 - `MYSQL_DATABASE: items`
 - `MYSQL_ROOT_PASSWORD: r00tpa55`

5.1. Cree e inicie el contenedor.

```
[student@workstation ~]$ sudo podman run --name mysql-2 \
> -d -v /var/local/mysql:/var/lib/mysql/data \
> -p 13306:3306 \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mpa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> rhscl/mysql-57-rhel7
281c0e2790e54cd5a0b8e2a8cb6e3969981b85cde8ac611bf7ea98ff78bdffbb
```

5.2. Verifique que el contenedor se haya iniciado correctamente.

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID      ...output omitted...      NAMES
281c0e2790e5      ...output omitted...      mysql-2
```

6. Guarde la lista de todos los contenedores (incluidos los detenidos) en el archivo `/tmp/my-containers`.

Guarde la información con el siguiente comando:

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

7. Acceda a la shell de Bash dentro del contenedor y verifique que la base de datos `items` y la tabla `Item` (ítem) aún estén disponibles. Además, confirme que la tabla contiene datos.

7.1. Acceda a la shell de Bash dentro del contenedor.

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

7.2. Conéctese al servidor MySQL.

```
bash-4.2$ mysql -uroot
```

- 7.3. Enumere todas las bases de datos y confirme que la base de datos `items` está disponible.

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

- 7.4. Enumere todas las tablas de la base de datos **items** y verifique que la tabla **Item** esté disponible.

```
mysql> use items;
Database changed
mysql> show tables;
+-----+
| Tables_in_items |
+-----+
| Item           |
+-----+
1 row in set (0.01 sec)
```

- 7.5. Vea los datos de la tabla.

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

- 7.6. Salga del cliente MySQL y de la shell del contenedor.

```
mysql> exit
Bye
bash-4.2$ exit
```

8. Mediante el reenvío de puertos, inserte una nueva fila en la tabla **Item**. La fila debe tener un valor de **description** de **Finished lab** (Trabajo de laboratorio finalizado) y un valor de **done** de **1**.

- 8.1. Conéctese con la base de datos MySQL.

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

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

MySQL [items]>
```

8.2. Inserte la nueva fila.

```
MySQL[items]> insert into Item (description, done) values ('Finished lab', 1);
Query OK, 1 row affected (0.00 sec)
```

8.3. Salga del cliente MySQL.

```
MySQL[items]> exit
Bye
```

9. Dado que ya no se necesita el primer contenedor, elimínelo para liberar recursos.

Use el comando siguiente para eliminar el contenedor:

```
[student@workstation ~]$ sudo podman rm mysql-1
616azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cdd
```

Evaluación

Evalúe su trabajo ejecutando el comando **lab manage-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el comando **lab manage-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Podman tiene subcomandos para: crear un nuevo contenedor (**run**), borrar un contenedor (**rm**), enumerar los contenedores (**ps**), detener un contenedor (**stop**) e iniciar un proceso en un contenedor (**exec**).
- El almacenamiento predeterminado del contenedor es efímero, lo que significa que su contenido no se conserva después de que se elimina o se reinicia el contenedor.
 - Los contenedores pueden usar una carpeta del sistema de archivos de host para trabajar con datos persistentes.
 - Podman monta volúmenes en un contenedor con la opción **-v** en el comando **podman run**.
- El comando **podman exec** inicia un proceso adicional dentro de un contenedor en ejecución.
- Podman asigna puertos locales a puertos de contenedores mediante el uso de la opción **-p** en el subcomando **run**.

capítulo 4

Administración de imágenes de contenedores

Meta

Administrar el ciclo de vida de la imagen de un contenedor desde su creación hasta su eliminación.

Objetivos

- Buscar imágenes en registros remotos y extraerlas.
- Exportar, importar y administrar imágenes de contenedores de forma local y en un registro.

Secciones

- Acceso a los registros (y cuestionario)
- Manipulación de imágenes de contenedores (y ejercicio guiado)

Trabajo de laboratorio

- Administración de imágenes de contenedores

Acceso a los registros

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Buscar y extraer imágenes de registros remotos con los comandos de Podman y la API REST del registro.
- Enumerar las ventajas de usar un registro público certificado para descargar imágenes seguras.
- Personalizar la configuración de Podman para acceder a registros de imágenes de contenedores alternativos.
- Enumerar imágenes descargadas de un registro en el sistema de archivos local.
- Administrar etiquetas para extraer imágenes etiquetadas.

Registros públicos

Los registros de imágenes son servicios que ofrecen imágenes de contenedor para descargar. Permiten que los creadores y mantenedores de imágenes almacenen y distribuyan imágenes de contenedor a audiencias públicas o privadas.

Podman busca y descarga imágenes de contenedores de registros públicos y privados. Red Hat Container Catalog es el registro público de imágenes administrado por Red Hat. Aloja un gran conjunto de imágenes de contenedores, incluidas aquellas provistas por los principales proyectos de código abierto, como Apache, MySQL y Jenkins. Todas las imágenes de Container Catalog son examinadas por el equipo de seguridad interno de Red Hat, lo que significa que son confiables y están protegidas contra fallas de seguridad.

Las imágenes de contenedores de Red Hat proporcionan los siguientes beneficios:

- *Fuente de confianza*: Todas las imágenes de contenedores provienen de fuentes conocidas y de confianza para Red Hat.
- *Dependencias originales*: Ninguno de los paquetes de contenedores se ha usado de forma indebida, y estos solo incluyen librerías conocidas.
- *Sin vulnerabilidades*: Las imágenes de contenedores están libres de vulnerabilidades conocidas en los componentes o las capas de la plataforma.
- *Protección de tiempo de ejecución* : Todas las aplicaciones en imágenes de contenedores se ejecutan con usuarios que no son root, lo que minimiza la superficie de exposición a aplicaciones maliciosas o defectuosas.
- *Compatible con Red Hat Enterprise Linux (RHEL)*: Las imágenes de contenedores son compatibles con todas las plataformas de RHEL, desde las físicas (bare-metal) hasta las basadas en la nube.
- *Soporte de Red Hat*: La pila (stack) completa cuenta con el soporte comercial de Red Hat.

Quay.io es otro repositorio de imágenes públicas patrocinado por Red Hat. Quay.io presenta varias funciones interesantes, como la creación de imágenes en el servidor, controles de acceso detallados y escaneo automático de imágenes para detectar vulnerabilidades conocidas.

Si bien las imágenes de Red Hat Container Catalog son de confianza y verificadas, Quay.io ofrece imágenes en vivo que los creadores actualizan con regularidad. Los usuarios de Quay.io pueden crear sus espacios de nombres, con un control de acceso detallado, y publicar las imágenes que crean en ese espacio de nombres. Los usuarios de Container Catalog rara vez o nunca envían nuevas imágenes, sino que consumen imágenes de confianza generadas por el equipo de Red Hat.

Registros privados

Algunos creadores o mantenedores de imágenes desean que sus imágenes sean públicas. Sin embargo, otros creadores de imágenes prefieren mantener sus imágenes privadas debido a:

- Disposiciones de privacidad y protección de secretos de la empresa.
- Restricciones legales y leyes.
- Evitar la publicación de imágenes en desarrollo.

Los registros privados les dan a los creadores de imágenes el control sobre la colocación, la distribución y el uso de sus imágenes.

Configuración de registros en Podman

Para configurar registros para el comando **podman**, debe actualizar el archivo **/etc/containers/registries.conf**. Edite la entrada de **registros** en la sección **[registries.search]**, agregando una entrada a la lista de valores.

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



nota

Use un FQDN y un número de puerto para identificar un registro. Un registro que no incluye un número de puerto tiene un número de puerto predeterminado de 5000. Si el registro usa un puerto diferente, debe especificarse. Indique los números de puerto agregando dos puntos (:) y el número de puerto después del FQDN.

Las conexiones seguras a un registro requieren un certificado de confianza. Para soportar conexiones inseguras, agregue el nombre de registro a la entrada **registries** en la sección **[registries.insecure]** del archivo **/etc/containers/registries.conf**:

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

Acceso a los registros

Búsqueda de imágenes en registros

El comando **podman search** busca imágenes por nombre de imagen, nombre de usuario o descripción de todos los registros enumerados en el archivo de configuración **/etc/containers/registries.conf**. La sintaxis para el comando **podman search** se muestra a continuación:

```
[student@workstation ~]$ sudo podman search [OPTIONS] <term>
```

En la siguiente tabla, se muestran algunas opciones útiles disponibles para el subcomando **search**:

Opción	Descripción
--limit <number>	Limita el número de imágenes enumeradas por registro.
--filter <filter=value>	Filtrá la salida en función de las condiciones proporcionadas. Estos son los filtros soportados: <ul style="list-style-type: none"> • stars=<number>: Mostrar solo imágenes con al menos esta cantidad de estrellas. • is-automated=<true false>: Mostrar solo las imágenes creadas automáticamente. • is-official=<true false>: Mostrar solo las imágenes marcadas como oficiales.
--tls-verify <true false>	Habilita o deshabilita la validación de certificados HTTPS para todos los registros usados. true

API HTTP de registro

Un registro remoto expone servicios web que proporcionan una interfaz de programación de aplicaciones (API) al registro. Podman usa estas interfaces para acceder e interactuar con repositorios remotos. Muchos registros cumplen con la especificación **Docker Registry HTTP API v2**, que expone una interfaz REST estandarizada para la interacción del registro. Puede usar esta interfaz REST para interactuar directamente con un registro, en lugar de usar Podman.

A continuación se presentan algunas muestras del uso de esta API con los comandos **curl**:

Para enumerar todos los repositorios disponibles en un registro, use el extremo **/v2/_catalog**. El parámetro **n** se usa para limitar el número de repositorios para mostrar.

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

**nota**

Si Python está disponible, úselo para formatear la respuesta de JSON:

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

El extremo **/v2/<name>/tags/list** proporciona la lista de etiquetas disponibles para una sola imagen:

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

**nota**

Quay.io ofrece una API dedicada para interactuar con los repositorios más allá de lo especificado en la API de repositorio de Docker. Vea <https://docs.quay.io/api/> para obtener más detalles.

Autenticación de registro

Algunos registros de imágenes de contenedores requieren autorización de acceso. El comando **podman login** permite la autenticación de nombre de usuario y contraseña en un registro:

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

La API HTTP de registro requiere credenciales de autenticación. Primero, use el servicio de inicio de sesión único (SSO) de Red Hat para obtener un token de acceso:

```
[student@workstation ~]$ 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...}[student@workstation ~]$
```

A continuación, incluya este token en un encabezado de autorización de **portador** en las solicitudes posteriores:

```
[student@workstation ~]$ curl -H "Authorization: Bearer eyJh...mgL4" \
> -Ls https://registry.redhat.io/v2/rhscl/mysql-57-rhel7/tags/list \
> | python -mjson.tool
{
  "name": "rhscl/mysql-57-rhel7",
  "tags": [
    "5.7-3.9",
    "5.7-3.8",
    "5.7-3.4",
    "5.7-3.7",
    ...output omitted...
```



nota

Otros registros pueden requerir pasos diferentes para proporcionar credenciales.

Si un registro cumple con **Docker Registry HTTP v2 API**, la autenticación se ajusta al esquema RFC7235.

Extracción de imágenes

Para extraer imágenes de contenedores desde un registro, use el comando **podman pull**:

```
[student@workstation ~]$ sudo podman pull [OPTIONS] [REGISTRY[:PORT]/]NAME[:TAG]
```

El comando **podman pull** usa el nombre de la imagen obtenido del subcomando **search** para extraer una imagen del registro. El subcomando **pull** permite agregar el nombre de registro a la imagen. Esta variante soporta tener la misma imagen en varios registros.

Por ejemplo, para extraer un contenedor NGINX del registro quay.io, use el siguiente comando:

```
[student@workstation ~]$ sudo podman pull quay.io/bitnami/nginx
```



nota

Si el nombre de la imagen no incluye un nombre de registro, Podman busca una imagen de contenedor que coincida con los registros enumerados en el archivo de configuración **/etc/containers/registries.conf**. Podman busca imágenes en registros en el mismo orden en que aparecen en el archivo de configuración.

Enumeración de copias locales de imágenes

Todas las imágenes de contenedores descargadas de un registro se almacenan localmente en el mismo host donde se ejecutó el comando **podman**. Este comportamiento evita la repetición de descargas de imágenes y minimiza el tiempo de implementación de un contenedor. Podman también almacena las imágenes de contenedores personalizadas que cree en el mismo almacenamiento local.



nota

De manera predeterminada, Podman almacena las imágenes de contenedores en el directorio **/var/lib/containers/storage/overlay-images**.

Podman proporciona un subcomando **images** para enumerar todas las imágenes de contenedores almacenadas localmente.

```
[student@workstation ~]$ sudo podman images
REPOSITORY           TAG      IMAGE ID      CREATED     SIZE
registry.redhat.io/rhscl/mysql-57-rhel7  latest  c07bf25398f4  13 days ago  444MB
```

Etiquetas de imágenes

Una etiqueta de imagen es un mecanismo para soportar varias versiones de la misma imagen. Esta función sirve cuando se proveen varias versiones del mismo software, como un contenedor listo para producción o las actualizaciones más recientes del mismo software desarrollado para la evaluación de la comunidad. Todos los subcomandos de Podman que requieren un nombre de imagen de contenedor aceptan un parámetro de etiqueta para diferenciar entre varias etiquetas. Si el nombre de una imagen no contiene una etiqueta, el valor predeterminado de la etiqueta es **latest**. Por ejemplo, para extraer una imagen con la etiqueta **5.7** de **rhscl/mysql-57-rhel7**, use el siguiente comando:

```
[student@workstation ~]$ sudo podman pull rhscl/mysql-57-rhel7:5.7
```

Para iniciar un nuevo contenedor basado en la imagen **rhscl/mysql-57-rhel7:5.7**, use el siguiente comando:

```
[student@workstation ~]$ sudo podman run rhscl/mysql-57-rhel7:5.7
```



Referencias

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: Autenticación

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

► Cuestionario

Trabajo con registros

Elija las respuestas correctas para las siguientes preguntas, según la siguiente información:

Podman está disponible en un host RHEL con la siguiente entrada en el archivo `/etc/containers/registries.conf`:

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

Los hosts `registry.redhat.io` y `quay.io` tienen un registro ejecutándose, ambos tienen certificados válidos y usan la versión 1 del registro. Las siguientes imágenes están disponibles para cada host:

Nombres/etiquetas de imagen por registro

Registro	Imagen
registry.redhat.io	<ul style="list-style-type: none"> • nginx/1.0 • mysql/5.6 • httpd/2.2
quay.io	<ul style="list-style-type: none"> • mysql/5.5 • httpd/2.4

No hay imágenes disponibles localmente.

► 1. ¿Cuáles son los dos comandos que muestran las imágenes `mysql` disponibles para descargar desde `registry.redhat.io`? (Elija dos opciones).

- `podman search registry.redhat.io/mysql`
- `podman images`
- `podman pull mysql`
- `podman search mysql`

► 2. ¿Qué comando se usa para enumerar todas las etiquetas de imágenes disponibles para la imagen de contenedor `httpd`?

- `podman search httpd`
- `podman images httpd`
- `podman pull --all-tags=true httpd`
- No hay un comando de Podman disponible para buscar etiquetas.

► 3. ¿Cuáles son los dos comandos para extraer la imagen httpd con la etiqueta 2.2? (Elija dos opciones).

- a. `podman pull httpd:2.2`
- b. `podman pull httpd:latest`
- c. `podman pull quay.io/httpd`
- d. `podman pull registry.redhat.io/httpd:2.2`

► 4. Al ejecutar los siguientes comandos, ¿qué imágenes de contenedores se descargarán?

```
podman pull registry.redhat.io/httpd:2.2  
podman pull quay.io/mysql:5.6
```

- a. `quay.io/httpd:2.2`
`registry.redhat.io/mysql:5.6`
- b. `registry.redhat.io/httpd:2.2`
`registry.redhat.io/mysql:5.6`
- c. `registry.redhat.io/httpd:2.2`
No se descargará ninguna imagen para mysql.
- d. `quay.io/httpd:2.2`
No se descargará ninguna imagen para mysql.

► Solución

Trabajo con registros

Elija las respuestas correctas para las siguientes preguntas, según la siguiente información:

Podman está disponible en un host RHEL con la siguiente entrada en el archivo `/etc/containers/registries.conf`:

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

Los hosts `registry.redhat.io` y `quay.io` tienen un registro ejecutándose, ambos tienen certificados válidos y usan la versión 1 del registro. Las siguientes imágenes están disponibles para cada host:

Nombres/etiquetas de imagen por registro

Registro	Imagen
registry.redhat.io	<ul style="list-style-type: none"> • nginx/1.0 • mysql/5.6 • httpd/2.2
quay.io	<ul style="list-style-type: none"> • mysql/5.5 • httpd/2.4

No hay imágenes disponibles localmente.

► 1. ¿Cuáles son los dos comandos que muestran las imágenes `mysql` disponibles para descargar desde `registry.redhat.io`? (Elija dos opciones).

- `podman search registry.redhat.io/mysql`
- `podman images`
- `podman pull mysql`
- `podman search mysql`

► 2. ¿Qué comando se usa para enumerar todas las etiquetas de imágenes disponibles para la imagen de contenedor `httpd`?

- `podman search httpd`
- `podman images httpd`
- `podman pull --all-tags=true httpd`
- No hay un comando de Podman disponible para buscar etiquetas.

► 3. ¿Cuáles son los dos comandos para extraer la imagen httpd con la etiqueta 2.2? (Elija dos opciones).

- a. **podman pull httpd:2.2**
- b. **podman pull httpd:latest**
- c. **podman pull quay.io/httpd**
- d. **podman pull registry.redhat.io/httpd:2.2**

► 4. Al ejecutar los siguientes comandos, ¿qué imágenes de contenedores se descargarán?

```
podman pull registry.redhat.io/httpd:2.2  
podman pull quay.io/mysql:5.6
```

- a. quay.io/httpd:2.2
registry.redhat.io/mysql:5.6
- b. registry.redhat.io/httpd:2.2
registry.redhat.io/mysql:5.6
- c. registry.redhat.io/httpd:2.2
No se descargará ninguna imagen para mysql.
- d. quay.io/httpd:2.2
No se descargará ninguna imagen para mysql.

Manipulación de imágenes de contenedores

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Guardar y cargar imágenes de contenedor en archivos locales.
- Eliminar imágenes del almacenamiento local.
- Crear nuevas imágenes de contenedor a partir de contenedores y actualizar los metadatos de la imagen.
- Administrar las etiquetas de las imágenes con fines de distribución.

Introducción

Hay diversas maneras de administrar contenedores de imágenes en conformidad con los principios de devops. Por ejemplo, un desarrollador termina de probar un contenedor personalizado en una máquina y necesita transferir esta imagen de contenedor a otro host para otro desarrollador o a un servidor de producción. Es posible hacer esto de dos maneras:

1. Guardar la imagen de contenedor en un archivo **.tar**.
2. Publicar (*enviar*) la imagen de contenedor en un registro de imágenes.



nota

Hablaremos sobre una de las maneras en la que un desarrollador pudo haber creado este contenedor personalizado más adelante en este capítulo (**podman commit**). Sin embargo, en los siguientes capítulos, analizamos la forma recomendada de hacerlo con **Dockerfiles**.

Guardado y carga de imágenes

Las imágenes existentes del almacenamiento local de Podman se pueden guardar en un archivo **.tar** con el comando **podman save**. El archivo generado no es una colección de archivos TAR normal; contiene metadatos de la imagen y conserva las capas de la imagen original. Con este archivo, Podman puede recrear la imagen original exactamente como estaba.

La sintaxis general del subcomando **save** es la siguiente:

```
[student@workstation ~]$ sudo podman save [-o FILE_NAME] IMAGE_NAME[:TAG]
```

Podman envía la imagen generada a la salida estándar como datos binarios. Para evitar eso, use la opción **-o**.

En el siguiente ejemplo, la imagen de contenedor de MySQL descargada anteriormente de Red Hat Container Catalog se guarda en el archivo **mysql.tar**:

```
[student@workstation ~]$ sudo podman save \  
> -o mysql.tar registry.access.redhat.com/rhscl/mysql-57-rhel7:5.7
```

Use los archivos **.tar** generados por el subcomando **save** con fines de copia de seguridad. Para restaurar la imagen de contenedor, use el comando **podman load**. La sintaxis general del comando es la siguiente:

```
[student@workstation ~]$ sudo podman load [-i FILE_NAME]
```

Por ejemplo, este comando cargaría una imagen guardada en un archivo llamado **mysql.tar**.

```
[student@workstation ~]$ sudo podman load -i mysql.tar
```

Si el archivo **.tar** enviado como argumento no es una imagen de contenedor con metadatos, el comando **podman load** fallará.



nota

Para ahorrar espacio en disco, comprima el archivo generado por el subcomando **save** con Gzip usando el parámetro **--compress**. El subcomando **load** usa el comando **gunzip** antes de importar el archivo al almacenamiento local.

Eliminación de imágenes

Podman mantiene todas las imágenes descargadas en su almacenamiento local, incluso las que actualmente no se usan en ningún contenedor. Sin embargo, las imágenes pueden volverse obsoletas y deberían reemplazarse posteriormente.



nota

Las actualizaciones de las imágenes en un registro no se actualizan automáticamente. La imagen debe eliminarse y, luego, extraerse nuevamente para garantizar que el almacenamiento local tenga la versión más reciente de una imagen.

Para eliminar una imagen del almacenamiento local, ejecute el comando **podman rmi**. La sintaxis para este comando es la siguiente:

```
[student@workstation ~]$ sudo podman rmi [OPTIONS] IMAGE [IMAGE...]
```

Se puede hacer referencia a una imagen usando su nombre o su ID con fines de eliminación. Podman no puede eliminar una imagen si los contenedores están usando esa imagen. Debe detener y eliminar todos los contenedores que usan esa imagen antes de eliminarla.

Para evitar esto, el subcomando **rmi** tiene la opción **--force**. Esta opción obliga a eliminar una imagen, incluso si la imagen se usa en varios contenedores o si estos contenedores se están ejecutando. Podman detiene y elimina todos los contenedores que usan la imagen eliminada a la fuerza antes de eliminarla.

Eliminación de todas las imágenes

Para eliminar todas las imágenes que no son usadas por ningún contenedor, use el siguiente comando:

```
[student@workstation ~]$ sudo podman rmi -a
```

El comando muestra todos los ID de imágenes disponibles en el almacenamiento local y los pasa como parámetros al comando **podman rmi** para su eliminación. Las imágenes que están en uso no se eliminan. Sin embargo, esto no impide que se eliminen las imágenes que no están en uso.

Modificación de imágenes

Lo ideal sería que todas las imágenes de contenedores se compilen con **Dockerfile** para generar un conjunto de capas de imágenes limpia y reducida, sin archivos de registro, archivos temporales u otros artefactos creados por la personalización del contenedor. Sin embargo, algunos usuarios pueden proporcionar imágenes de contenedores como están, sin un **Dockerfile**. Como enfoque alternativo para la creación de nuevas imágenes, modifique un contenedor en ejecución en el lugar y guarde sus capas para crear una nueva imagen de contenedor. El comando **podman commit** proporciona esta función.



Advertencia

Si bien el comando **podman commit** es el enfoque más directo para crear nuevas imágenes, este no se recomienda, debido al tamaño de la imagen (**commit** guarda los registros y archivos de ID de proceso en las capas capturadas), y a la falta de trazabilidad de los cambios. Un **Dockerfile** proporciona un mecanismo sólido para personalizar e implementar cambios en un contenedor con un conjunto de comandos legible por los humanos sin el conjunto de archivos que genera el sistema operativo.

La sintaxis para el comando **podman commit** es la siguiente:

```
[student@workstation ~]$ sudo podman commit [OPTIONS] CONTAINER \
> [REPOSITORY[:PORT]/]IMAGE_NAME[:TAG]
```

En la siguiente tabla, se muestran las opciones más importantes disponibles para el comando **podman commit**:

Opción	Descripción
--author ""	Identifica quién creó la imagen de contenedor.
--message ""	Incluye un mensaje de confirmación en el registro.
--format	Selecciona el formato de la imagen. Las opciones válidas son oci y docker .



nota

La opción **--message** no está disponible en el formato de contenedor OCI predeterminado.

Para encontrar el ID de un contenedor en ejecución en Podman, ejecute el comando **podman ps**:

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID IMAGE ... NAMES
87bdfcc7c656 mysql ...output omitted... mysql-basic
```

Con el tiempo, los administradores pueden personalizar la imagen y establecer el contenedor en el estado deseado. Para identificar los archivos que se modificaron, se crearon o se eliminaron desde que se inició el contenedor, use el subcomando **diff**. Este subcomando solo requiere el nombre o ID del contenedor:

```
[student@workstation ~]$ sudo 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
```

El subcomando **diff** etiqueta cualquier archivo agregado con una **A**, cualquier archivo modificado con una **C** y cualquier archivo eliminado con una **D**.



nota

El comando **diff** solo informa los archivos agregados, modificados o eliminados al sistema de archivos del contenedor. Los archivos montados en un contenedor en ejecución no se consideran parte del sistema de archivos del contenedor.

Use el comando **podman inspect** para recuperar la lista de archivos y directorios montados para un contenedor en ejecución:

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

No se muestra ningún archivo de esta lista, ni archivo en un directorio de esta lista, en el resultado del comando **podman diff**.

Para confirmar los cambios a otra imagen, ejecute el siguiente comando:

```
[student@workstation ~]$ sudo podman commit mysql-basic mysql-custom
```

Etiquetado de imágenes

Un proyecto con varias imágenes basadas en el mismo software podría distribuirse y crear proyectos individuales para cada imagen, pero este enfoque requiere mantenimiento adicional para administrar e implementar las imágenes en las ubicaciones correctas.

Los registros de imágenes de contenedores soportan etiquetas para distinguir varias versiones del mismo proyecto. Por ejemplo, un cliente puede usar una imagen de contenedor para ejecutar con una base de datos MySQL o PostgreSQL, con una etiqueta como manera para diferenciar la base de datos que usará una imagen de contenedor.

**nota**

Generalmente, los desarrolladores de contenedores usan las etiquetas para distinguir entre varias versiones del mismo software. Se proporcionan varias etiquetas para identificar una versión con facilidad. El sitio web oficial de imágenes de contenedores de MySQL usa la versión como nombre de la etiqueta (**5.5.16**). Además, la misma imagen tiene una segunda etiqueta con la versión secundaria, por ejemplo 5.5, para minimizar la necesidad de obtener la versión más reciente para una versión específica.

Para etiquetar una imagen, use el comando **podman tag**:

```
[student@workstation ~]$ sudo podman tag [OPTIONS] IMAGE[:TAG] \
> [REGISTRYHOST/] [USERNAME/]NAME[:TAG]
```

El argumento **IMAGE** es el nombre de la imagen con una etiqueta opcional administrada por Podman. El siguiente argumento se refiere al nuevo nombre alternativo para la imagen. Podman supone que se usa la última versión, como lo indica la etiqueta **latest**, si no figura el valor de la etiqueta. Por ejemplo, para etiquetar una imagen, use el siguiente comando:

```
[student@workstation ~]$ sudo podman tag mysql-custom devops/mysql
```

La opción **mysql-custom** corresponde al nombre de la imagen en el registro de contenedores.

Para usar un nombre de etiqueta diferente, use el siguiente comando en su lugar:

```
[student@workstation ~]$ sudo podman tag mysql-custom devops/mysql:snapshot
```

Eliminación de etiquetas de las imágenes

Se pueden asignar varias etiquetas a una sola imagen con el comando **podman tag**. Para eliminarlas, use el comando **podman rmi**, como se mencionó anteriormente:

```
[student@workstation ~]$ sudo podman rmi devops/mysql:snapshot
```

**nota**

Dado que varias etiquetas pueden apuntar a la misma imagen, para eliminar una imagen a la que hacen referencia varias etiquetas, primero se debe eliminar cada etiqueta de forma individual.

Prácticas recomendadas para etiquetar imágenes

Podman agrega automáticamente la etiqueta **latest** si no se especifica ninguna etiqueta, ya que Podman considera que la imagen es la compilación más reciente. Sin embargo, esto puede no ser cierto según el modo en que se usan las etiquetas en cada proyecto. Por ejemplo, la mayoría de los proyectos de código abierto considera que la etiqueta **latest** coincide con la versión más reciente, no la compilación más reciente.

Además, se proporcionan varias etiquetas para minimizar la necesidad de recuperar la versión más reciente de una determinada versión de un proyecto. Por lo tanto, si hay una versión de un

proyecto (por ejemplo, **2.1.10**), se puede crear otra etiqueta denominada **2.1** que apunte a la misma imagen de la versión **2.1.10**. Esto simplifica la extracción de imágenes del registro.

Publicación de imágenes en un registro

Para publicar una imagen en un registro, debe residir en el almacenamiento local de Podman y etiquetarse con fines de identificación. Para enviar la imagen al registro, la sintaxis del subcomando **push** es:

```
[student@workstation ~]$ sudo podman push [OPTIONS] IMAGE [DESTINATION]
```

Por ejemplo, para enviar la imagen **bitnami/nginx** a su repositorio, use el siguiente comando:

```
[student@workstation ~]$ sudo podman push quay.io/bitnami/nginx
```



Referencias

Sitio de Podman

<https://podman.io/>

► Ejercicio Guiado

Creación de una imagen de contenedor de Apache personalizada

En este ejercicio guiado, creará una imagen de contenedor de Apache personalizada con el comando **podman commit**.

Resultados

Debería poder crear una imagen de contenedor personalizada.

Antes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

```
[student@workstation ~]$ lab image-operations start
```

- ▶ 1. Abra un terminal en **workstation** (**Applications [Aplicaciones]** → **System Tools [Herramientas del sistema]** → **Terminal**). Inicie sesión en su cuenta de Quay.io e inicie un contenedor mediante el uso de la imagen disponible en **quay.io/redhattraining/httpd-parent**. La opción **-p** le permite especificar un puerto de redirecciónamiento. En este caso, Podman reenvía las solicitudes entrantes del puerto 8180 TCP del host al puerto 80 TCP del contenedor.

```
[student@workstation ~]$ sudo podman login quay.io
Username: your_quay_username
Password: your_quay_password
Login Succeeded!
[student@workstation ~]$ sudo podman run -d --name official-httpd \
> -p 8180:80 redhattraining/httpd-parent
...output omitted...
Writing manifest to image destination
Storing signatures
3a6baecaff2b4e8c53b026e04847dda5976b773ade1a3a712b1431d60ac5915d
```

Su última línea del resultado es diferente de la última línea que se muestra arriba. Observe los primeros doce caracteres.

- ▶ 2. Cree una página HTML en el contenedor **official-httpd**.

2.1. Acceda a la shell del contenedor con el subcomando **exec** y cree una página HTML.

```
[student@workstation ~]$ sudo podman exec -it official-httpd /bin/bash
bash-4.4# echo "DO180 Page" > /var/www/html/do180.html
```

2.2. Salga del contenedor.

```
bash-4.4# exit
```

- 2.3. Asegúrese de que se pueda acceder al archivo HTML desde la máquina virtual **workstation** con el comando **curl**.

```
[student@workstation ~]$ curl 127.0.0.1:8180/do180.html
DO180 Page
```

Debe ver la siguiente salida:

```
DO180 Page
```

- 3. Use el subcomando **diff** para examinar las diferencias en el contenedor entre la imagen y la nueva capa creada por el contenedor.
- 3.1. Recupere la lista de archivos y directorios externos que Podman monta en el contenedor en ejecución:

```
[student@workstation ~]$ sudo podman inspect -f \
> "{{range .Mounts}}{{println .Destination}}{{end}}" official-httdp
/proc
/dev
/dev/pts
/dev/shm
/dev/mqueue
/sys
/sys/fs/cgroup
```

- 3.2. Ejecute el comando **podman diff** para ver una lista de archivos modificados en el sistema de archivos del contenedor:

```
[student@workstation ~]$ sudo podman diff official-httdp
C /etc
C /root
A /root/.bash_history
...output omitted...
C /tmp
C /var
C /var/log
C /var/log/httpd
A /var/log/httpd/access_log
A /var/log/httpd/error_log
C /var/www
C /var/www/html
A /var/www/html/do180.html
```

Observe que ninguno de los archivos o directorios en el resultado se encuentran en la lista de archivos montados de forma externa en el paso anterior.

**nota**

A menudo, las imágenes del contenedor del servidor web etiquetan el directorio **/var/www/html** como volumen. En estos casos, los archivos agregados a este directorio no se consideran parte del sistema de archivos del contenedor y no se mostrarán en el resultado del comando **git diff**.

La imagen del contenedor **redhattraining/httpd-parent** no etiqueta el directorio **/var/www/html** como volumen. Como resultado, la modificación del archivo **/var/www/html/do180.html** se considera una modificación al sistema de archivos del contenedor subyacente.

- 4. Cree una nueva imagen con los cambios creados por el contenedor en ejecución.

- 4.1. Detenga el contenedor **official-httdp**.

```
[student@workstation ~]$ sudo podman stop official-httdp
3a6baecaff2b4e8c53b026e04847dda5976b773ade1a3a712b1431d60ac5915d
```

- 4.2. Confirme los cambios realizados en la nueva imagen de contenedor con un nombre nuevo. Use su nombre como el autor de los cambios.

```
[student@workstation ~]$ sudo 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
31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb399befef2a23
```

- 4.3. Enumere las imágenes de contenedores disponibles.

```
[student@workstation ~]$ sudo podman images
```

La salida esperada es similar a la siguiente:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/do180-custom-httdp	latest	31c3ac78e9d4
quay.io/redhattraining/httpd-parent	latest	2cc07fbb5000

El ID de la imagen coincide con los primeros 12 caracteres del hash. Las imágenes más recientes aparecen en la parte superior.

- 5. Publique la imagen de contenedor guardada en el registro de contenedores.

- 5.1. Cargue la configuración de su entorno de aula.

capítulo 4 | Administración de imágenes de contenedores

Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 5.2. Para etiquetar la imagen con el nombre de host y la etiqueta del registro, ejecute el siguiente comando.

```
[student@workstation ~]$ sudo podman tag do180-custom-httd \> quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httd:v1.0
```

- 5.3. Ejecute el comando **podman images** para garantizar que el nuevo nombre se haya agregado a la caché.

```
[student@workstation ~]$ sudo podman images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/do180-custom-httd	latest	31c3ac78e9d4
quay.io/your_quay_username/do180-custom-httd	v1.0	31c3ac78e9d4
quay.io/redhattraining/httpd-parent	latest	2cc07fbb5000

- 5.4. Publique la imagen en su registro de Quay.io.

```
[student@workstation ~]$ sudo podman push \> quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httd:v1.0
```

Getting image source signatures

Copying blob sha256:071d8bd765171080d01682844524be57ac9883e53079b6ac66707e19...
200.44 MB / 200.44 MB [=====] 1m38s
...output omitted...

Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeab3...
2.99 KB / 2.99 KB [=====] 0s

Writing manifest to image destination

Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeab3...
0 B / 2.99 KB [-----] 0s

Writing manifest to image destination

Storing signatures

**nota**

Al enviar la imagen **do180-custom-httd**, se crea un repositorio privado homónimo en su cuenta de Quay.io. Actualmente, los planes gratuitos de Quay.io no permiten repositorios privados. Puede crear el repositorio público antes de enviar la imagen o cambiar el repositorio a público más tarde.

- 5.5. Verifique que la imagen esté disponible en Quay.io. El comando **podman search** requiere que la imagen sea indexada por Quay.io. Esto puede demorar algunas horas; por lo tanto, use el comando **podman pull** para obtener la imagen. Esto demuestra la disponibilidad de la imagen.

```
[student@workstation ~]$ sudo podman pull \
> -q quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb3
```

- 6. Cree un contenedor de la imagen recientemente publicada.

Use el comando **podman run** para iniciar un nuevo contenedor. Use ***your_quay_username/do180-custom-httdp:v1.0*** como imagen base.

```
[student@workstation ~]$ sudo podman run -d --name test-httdp -p 8280:80 \
> ${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
c0f04e906bb12bd0e514cbd0e581d2746e04e44a468dfbc85bc29ffcc5acd16c
```

- 7. Use el comando **curl** para acceder a la página HTML. Asegúrese de usar el puerto 8280.

Se debería mostrar la página HTML creada en el paso anterior.

```
[student@workstation ~]$ curl http://localhost:8280/do180.html
DO180 Page
```

Finalizar

En **workstation**, ejecute el script **lab image-operations finish** para terminar este trabajo de laboratorio.

```
[student@workstation ~]$ lab image-operations finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Administración de imágenes

Listado de verificación de rendimiento

En este trabajo de laboratorio, creará y administrará imágenes de contenedores.

Resultados

Debería poder crear una imagen de contenedor personalizada y administrar imágenes de contenedores.

Antes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Use el comando **podman search** para localizar la imagen oficial **docker.io/nginx** y extraerla al sistema de archivos local.
Asegúrese de que la imagen se haya recuperado correctamente.
2. Inicie un nuevo contenedor con la imagen Nginx, según las especificaciones indicadas en la siguiente lista.
 - **Nombre:** **official-nginx**
 - **Ejecutar como daemon:** sí
 - **Imagen de contenedor:** **nginx**
 - **Reenvío de puertos:** desde el puerto 8080 del host hasta el puerto 80 del contenedor.
3. Inicie sesión en el contenedor con el subcomando **exec**. Reemplace el contenido del archivo **index.html** con **D0180**. El directorio del servidor web está ubicado en **/usr/share/nginx/html**.
Una vez actualizado el archivo, salga del contenedor y use el comando **curl** para acceder a la página web.
4. Detenga el contenedor en ejecución y confirme los cambios para crear una nueva imagen de contenedor. Asigne a la nueva imagen el nombre **do180/mynginx** y la etiqueta **v1.0-SNAPSHOT**. Use las siguientes especificaciones:
 - Nombre de la imagen: **do180/mynginx**
 - Etiqueta de la imagen: **v1.0-SNAPSHOT**
 - Nombre del autor: *su nombre*
5. Inicie un nuevo contenedor con la imagen Nginx actualizada, según las especificaciones indicadas en la siguiente lista.
 - **Nombre:** **official-nginx-dev**

- **Ejecutar como daemon:** sí
 - **Imagen de contenedor:** **do180/mynginx:v1.0-SNAPSHOT**
 - **Reenvío de puertos:** desde el puerto 8080 del host hasta el puerto 80 del contenedor.
6. Inicie sesión en el contenedor con el subcomando **exec** para hacer un último cambio. Reemplace el contenido del archivo **/usr/share/nginx/html/index.html** con **DO180 Page**.
- Una vez actualizado el archivo, salga del contenedor y use el comando **curl** para verificar los cambios.
7. Detenga el contenedor en ejecución y confirme los cambios para crear la última imagen de contenedor. Asígnele a la nueva imagen el nombre **do180/mynginx** y la etiqueta **v1.0**. Use las siguientes especificaciones:
- Nombre de la imagen: **do180/mynginx**
 - Etiqueta de la imagen: **v1.0**
 - Nombre del autor: *su nombre*
8. Elimine la imagen de desarrollo **do180/mynginx:v1.0-SNAPSHOT** del almacenamiento local de imágenes.
9. Use la imagen etiquetada **do180/mynginx:v1.0** para crear un nuevo contenedor con las siguientes especificaciones:
- Nombre del contenedor: **my-nginx**
 - Ejecutar como daemon: sí
 - Imagen de contenedor: **do180/mynginx:v1.0**
 - Reenvío de puertos: desde el puerto 8280 del host al puerto 80 del contenedor

En **workstation**, use el comando **curl** para acceder al servidor web, al que se puede acceder desde el puerto 8280.

Evaluación

Evalúe su trabajo ejecutando el comando **lab image-review grade** en su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el comando **lab image-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

► Solución

Administración de imágenes

Lista de verificación de rendimiento

En este trabajo de laboratorio, creará y administrará imágenes de contenedores.

Resultados

Debería poder crear una imagen de contenedor personalizada y administrar imágenes de contenedores.

Antes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Use el comando **podman search** para localizar la imagen oficial **docker.io/nginx** y extraerla al sistema de archivos local.
Asegúrese de que la imagen se haya recuperado correctamente.
 - 1.1. Use el comando **podman search** para buscar la imagen oficial **docker.io/nginx** del contenedor Nginx.

```
[student@workstation ~]$ sudo podman search docker.io/nginx
> -f is-official=true
INDEX          NAME                  DESCRIPTION              STARS  OFFICIAL...
docker.io      docker.io/library/nginx  Official build of Nginx.  12022  [OK]    ...
```

- 1.2. Use el comando **podman pull** para extraer la imagen de contenedor Nginx.

```
[student@workstation ~]$ sudo podman pull docker.io/nginx:1.17
Trying to pull Trying to pull docker.io/nginx:1.17...
...output omitted...
Storing signatures
42b4762643dcc9bf492b08064b55fef64942f055f0da91289a8abf93c6d6b43c
```

- 1.3. Ejecute el comando **podman images** para asegurarse de que la imagen de contenedor esté disponible a nivel local.

```
[student@workstation ~]$ sudo podman images
```

Este comando produce una salida similar a la siguiente:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
docker.io/library/nginx	1.17	42b4762643dc	8 days ago	130MB

2. Inicie un nuevo contenedor con la imagen Nginx, según las especificaciones indicadas en la siguiente lista.
- **Nombre:** `official-nginx`
 - **Ejecutar como daemon:** sí
 - **Imagen de contenedor:** `nginx`
 - **Reenvío de puertos:** desde el puerto 8080 del host hasta el puerto 80 del contenedor.
- 2.1. En `workstation`, use el comando `podman run` para crear un contenedor denominado `official-nginx`.

```
[student@workstation ~]$ sudo podman run --name official-nginx \
> -d -p 8080:80 docker.io/nginx:1.17
02dbc348c7dcf8560604a44b11926712f018b0ac44063d34b05704fb8447316f
```

3. Inicie sesión en el contenedor con el subcomando `exec`. Reemplace el contenido del archivo `index.html` con **D0180**. El directorio del servidor web está ubicado en `/usr/share/nginx/html`.
- Una vez actualizado el archivo, salga del contenedor y use el comando `curl` para acceder a la página web.
- 3.1. Inicie sesión en el contenedor con el comando `podman exec`.

```
[student@workstation ~]$ sudo podman exec -it official-nginx /bin/bash
root@02dbc348c7dc:/#
```

- 3.2. Actualice el archivo `index.html` ubicado en `/usr/share/nginx/html`. El archivo debería leer **D0180**.

```
root@02dbc348c7dc:/# echo 'D0180' > /usr/share/nginx/html/index.html
```

- 3.3. Salga del contenedor.

```
root@02dbc348c7dc:/# exit
```

- 3.4. Use el comando `curl` para asegurarse de que el archivo `index.html` está actualizado.

```
[student@workstation ~]$ curl 127.0.0.1:8080
D0180
```

4. Detenga el contenedor en ejecución y confirme los cambios para crear una nueva imagen de contenedor. Asígnele a la nueva imagen el nombre **do180/mynginx** y la etiqueta **v1.0-SNAPSHOT**. Use las siguientes especificaciones:
- Nombre de la imagen: **do180/mynginx**
 - Etiqueta de la imagen: **v1.0-SNAPSHOT**
 - Nombre del autor: *su nombre*

- 4.1. Use el comando **sudo podman stop** para detener el contenedor **official-nginx**.

```
[student@workstation ~]$ sudo podman stop official-nginx  
2dbc348c7dcf8560604a44b11926712f018b0ac44063d34b05704fb8447316f
```

- 4.2. Confirme los cambios para una nueva imagen de contenedor. Use su nombre como el autor de los cambios.

```
[student@workstation ~]$ sudo podman commit -a 'Your Name' \  
> official-nginx do180/mynginx:v1.0-SNAPSHOT  
Getting image source signatures  
...output omitted...  
Storing signatures  
4a13dd08d175a6095e6462e52431be1577ca931fc1aea139b71346bfc7f9c76
```

- 4.3. Enumere las imágenes de contenedores disponibles para buscar la imagen que acaba de crear.

```
[student@workstation ~]$ sudo podman images  
REPOSITORY           TAG      IMAGE ID   CREATED    SIZE  
localhost/do180/mynginx   v1.0-SNAPSHOT 4a13dd08d175  5 minutes ago  130MB  
docker.io/nginx        1.17      42b4762643dc  8 days ago   113MB
```

5. Inicie un nuevo contenedor con la imagen Nginx actualizada, según las especificaciones indicadas en la siguiente lista.

- **Nombre:** **official-nginx-dev**
- **Ejecutar como daemon:** sí
- **Imagen de contenedor:** **do180/mynginx:v1.0-SNAPSHOT**
- **Reenvío de puertos:** desde el puerto 8080 del host hasta el puerto 80 del contenedor.

- 5.1. En **workstation**, use el comando **podman run** para crear un contenedor denominado **official-nginx-dev**.

```
[student@workstation ~]$ sudo podman run --name official-nginx-dev \  
> -d -p 8080:80 do180/mynginx:v1.0-SNAPSHOT  
02dbc348c7dcf8560604a44b11926712f018b0ac44063d34b05704fb8447316f
```

6. Inicie sesión en el contenedor con el subcomando **exec** para hacer un último cambio. Reemplace el contenido del archivo **/usr/share/nginx/html/index.html** con **DO180 Page**. Una vez actualizado el archivo, salga del contenedor y use el comando **curl** para verificar los cambios.

- 6.1. Inicie sesión en el contenedor con el comando **podman exec**.

```
[student@workstation ~]$ sudo podman exec -it official-nginx-dev /bin/bash  
root@02dbc348c7dc:/#
```

capítulo 4 | Administración de imágenes de contenedores

- 6.2. Actualice el archivo **index.html** ubicado en **/usr/share/nginx/html**. El archivo debe indicar **DO180 Page**.

```
root@02dbc348c7dc:/# echo 'DO180 Page' > /usr/share/nginx/html/index.html
```

- 6.3. Salga del contenedor.

```
root@02dbc348c7dc:/# exit
```

- 6.4. Use el comando **curl** para asegurarse de que el archivo **index.html** está actualizado.

```
[student@workstation ~]$ curl 127.0.0.1:8080  
DO180 Page
```

7. Detenga el contenedor en ejecución y confirme los cambios para crear la última imagen de contenedor. Asígnele a la nueva imagen el nombre **do180/mynginx** y la etiqueta **v1.0**. Use las siguientes especificaciones:

- Nombre de la imagen: **do180/mynginx**
- Etiqueta de la imagen: **v1.0**
- Nombre del autor: *su nombre*

- 7.1. Use el comando **sudo podman stop** para detener el contenedor **official-nginx-dev**.

```
[student@workstation ~]$ sudo podman stop official-nginx-dev  
2dbc348c7dcf8560604a44b11926712f018b0ac44063d34b05704fb8447316f
```

- 7.2. Confirme los cambios para una nueva imagen de contenedor. Use su nombre como el autor de los cambios.

```
[student@workstation ~]$ sudo podman commit -a 'Your Name' \  
> official-nginx-dev do180/mynginx:v1.0  
Getting image source signatures  
...output omitted...  
Storing signatures  
4a13dd08d175a6095e6462e52431be1577ca931fcda1aea139b71346bfc7f9c76
```

- 7.3. Enumere las imágenes de contenedores disponibles para buscar la imagen que acaba de crear.

```
[student@workstation ~]$ sudo podman images  
REPOSITORY TAG IMAGE ID CREATED SIZE  
localhost/do180/mynginx v1.0 892569a87e3f 7 seconds ago 130MB  
localhost/do180/mynginx v1.0-SNAPSHOT 0857d81f5a4b 4 minutes ago 130MB  
docker.io/nginx 1.17 42b4762643dc 8 days ago 130MB
```

8. Elimine la imagen de desarrollo **do180/mynginx:v1.0-SNAPSHOT** del almacenamiento local de imágenes.

capítulo 4 | Administración de imágenes de contenedores

- 8.1. A pesar de estar detenido, el contenedor **official-nginx-dev** sigue presente. Muestre el contenedor con el comando **podman ps** con el indicador **-a**.

```
[student@workstation ~]$ sudo podman ps -a \
> --format="{{.ID}} {{.Names}} {{.Status}}"
e169c5fc8c3e    official-nginx-dev    Exited (0) 9 minutes ago
ccf046c2f87d    official-nginx        Exited (0) 12 minutes ago
```

- 8.2. Elimine el contenedor con el comando **podman rm**.

```
[student@workstation ~]$ sudo podman rm official-nginx-dev
e169c5fc8c3ed5c024af94aec752fa565650f9d07b95bb009329874801d859a10
```

- 8.3. Verifique que el contenedor se haya eliminado reenviando el mismo comando **podman ps**.

```
[student@workstation ~]$ sudo podman ps -a \
> --format="{{.ID}} {{.Names}} {{.Status}}"
ccf046c2f87d    official-nginx        Exited (0) 12 minutes ago
```

- 8.4. Use el comando **sudo podman rmi** para eliminar la imagen **do180/mynginx:v1.0-SNAPSHOT**.

```
[student@workstation ~]$ sudo podman rmi do180/mynginx:v1.0-SNAPSHOT
Untagged: localhost/do180/mynginx:v1.0-SNAPSHOT
```

- 8.5. Verifique que la imagen ya no esté presente al enumerar todas las imágenes con el comando **podman images**.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/do180/mynginx	v1.0	892569a87e3f	13 minutes ago	113MB
docker.io/library/nginx	1.17	42b4762643dc	8 days ago	130MB

9. Use la imagen etiquetada **do180/mynginx:v1.0** para crear un nuevo contenedor con las siguientes especificaciones:

- Nombre del contenedor: **my-nginx**
- Ejecutar como daemon: sí
- Imagen de contenedor: **do180/mynginx:v1.0**
- Reenvío de puertos: desde el puerto 8280 del host al puerto 80 del contenedor

En **workstation**, use el comando **curl** para acceder al servidor web, al que se puede acceder desde el puerto 8280.

- 9.1. Use el comando **sudo podman run** para crear el contenedor **my-nginx**, según las especificaciones.

```
[student@workstation ~]$ sudo podman run -d --name my-nginx \
> -p 8280:80 do180/mynginx:v1.0
c1cba44fa67bf532d6e661fc5e1918314b35a8d46424e502c151c48fb5fe6923
```

- 9.2. Use el comando **curl** para asegurarse de que la página **index.html** está disponible y devuelve el contenido personalizado.

```
[student@workstation ~]$ curl 127.0.0.1:8280
DO180 Page
```

Evaluación

Evalúe su trabajo ejecutando el comando **lab image-review grade** en su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el comando **lab image-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Red Hat Container Catalog proporciona imágenes probadas y certificadas en `registry.redhat.io`.
- Podman puede interactuar con registros de contenedores remotos para buscar, extraer y enviar imágenes de contenedores.
- Las etiquetas de imagen son un mecanismo para soportar varias versiones de una imagen de contenedor.
- Podman proporciona comandos para administrar imágenes de contenedores tanto en el almacenamiento local como en archivos comprimidos.
- Use el comando **podman commit** para crear una imagen desde un contenedor.

capítulo 5

Creación de imágenes de contenedores personalizadas

Meta

Diseñar y codificar un Dockerfile para compilar una imagen de contenedor personalizada.

Objetivos

- Describir los enfoques para crear imágenes de contenedores personalizadas.
- Crear una imagen de contenedor con comandos Dockerfile comunes.

Secciones

- Diseño de imágenes de contenedores personalizadas (y cuestionario)
- Compilación de imágenes de contenedores personalizadas con Dockerfiles (y ejercicio guiado)

Trabajo de laboratorio

- Creación de imágenes de contenedores personalizadas

Diseño de imágenes de contenedores personalizadas

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Describir los enfoques para crear imágenes de contenedores personalizadas.
- Buscar Dockerfiles existentes para usar como punto de partida a fin de crear una imagen de contenedor personalizada.
- Definir el rol de la librería Red Hat Software Collections Library (RHSCl) en el diseño de imágenes de contenedores desde el registro de Red Hat.
- Describir la alternativa de fuente a imagen (Source-to-Image, S2I) para Dockerfiles.

Reutilización de Dockerfiles existentes

Hasta ahora se ha cubierto un método para crear imágenes de contenedores: crear un contenedor, modificarlo para cumplir con los requisitos de la aplicación para ejecutar en él y, luego, confirmar los cambios en una imagen. Esta opción, si bien sencilla, solo es adecuada para usar o probar cambios muy específicos. No sigue las mejores prácticas de software, como mantenimiento, automatización de la creación y capacidad de repetición.

Los Dockerfiles son otra opción para crear imágenes de contenedor, a fin de abordar estas limitaciones. Los Dockerfiles son fáciles de compartir, controlar, reutilizar y ampliar.

Los Dockerfiles también facilitan la extensión de una imagen, denominada *imagen secundaria*, de otra imagen, denominada *imagen principal*. Una imagen secundaria incorpora todo en la imagen principal, así como todos los cambios y las adiciones realizados para crearla.

Para compartir y reutilizar imágenes, muchas aplicaciones, idiomas y marcos (frameworks) populares ya están disponibles en los registros de imágenes públicas, como Quay.io. No es trivial personalizar la configuración de una aplicación para seguir prácticas recomendadas para contenedores; de este modo, comenzar a partir de una imagen principal comprobada en general ahorra mucho trabajo.

El uso de imágenes principales de alta calidad mejora el mantenimiento, en especial, si la imagen principal se mantiene actualizada por su autor para dar cuenta de las correcciones de errores y los problemas de seguridad.

Los escenarios típicos para crear un Dockerfile para compilar una imagen secundaria de una imagen de contenedor existente incluyen los siguientes:

- Agregar nuevas librerías de tiempo de ejecución, como conectores de bases de datos.
- Incluir personalización en toda la organización, como certificados SSL y proveedores de autenticación.
- Agregar librerías internas, que se compartirán como una sola capa de imagen entre varias imágenes de contenedores para diferentes aplicaciones.

Cambiar un Dockerfile existente para crear una nueva imagen también puede ser un enfoque sensato en otros escenarios. Por ejemplo:

- Recortar la imagen de contenedor mediante la eliminación de material sin usar (como páginas del manual, o documentación que se encuentra en `/usr/share/doc`).
- Bloquear la imagen principal o algún paquete de software incluido para una versión específica a fin de reducir el riesgo relacionado con futuras actualizaciones de software.

Dos fuentes de imágenes de contenedores para usar como imágenes principales o para cambiar sus Dockerfiles son Docker Hub y la librería Red Hat Software Collections Library (RHSCl).

Trabajo con la librería Red Hat Software Collections Library

Red Hat Software Collections Library (RHSCl), o simplemente Software Collections, es una solución de Red Hat para desarrolladores que necesitan usar las herramientas de desarrollo más recientes que generalmente no se adaptan al calendario de versiones estándares de RHEL.

Red Hat Enterprise Linux (RHEL) proporciona un entorno estable para aplicaciones empresariales. Esto requiere que RHEL mantenga las principales versiones de paquetes upstream en el mismo nivel para evitar cambios de formato del archivo de configuración y de la API. Las correcciones de seguridad y rendimiento se replican de versiones upstream posteriores, pero no se migran las nuevas funciones que podrían perjudicar la compatibilidad con versiones anteriores.

RHSCl permite a los desarrolladores de software usar la versión más reciente sin afectar a RHEL, ya que los paquetes de RHSCl no reemplazan ni entran en conflicto con los paquetes de RHEL predeterminados. Los paquetes de RHEL y los paquetes de RHSCl predeterminados se instalan en paralelo.



nota

Todos los suscriptores de RHEL tienen acceso a la RHSCl. Para habilitar una colección de software particular para un usuario o un entorno de aplicación específicos (por ejemplo, MySQL 5.7, que se denomina `rh-mysql57`), habilite los repositorios Yum del software de RHSCl y siga unos simples pasos.

Búsqueda de Dockerfiles desde la librería Red Hat Software Collections Library

RHSCl es la fuente de la mayoría de las imágenes de contenedores provistas por el registro de imágenes de Red Hat para que usen los clientes de RHEL Atomic Host y OpenShift Container Platform.

Red Hat proporciona los Dockerfiles de RHSCl y las fuentes relacionadas en el paquete `rhscl-dockerfiles` disponible desde el repositorio de RHSCl. Los usuarios de la comunidad pueden obtener los Dockerfiles para imágenes de contenedores equivalentes basadas en CentOS de <https://github.com/sclorg?q=-container>.



nota

Muchas imágenes de contenedores RHSCl incluyen soporte para *Fuente a Imagen* (*Source-to-Image*, S2I), más conocido como una función de OpenShift Container Platform. Tener soporte para S2I no afecta el uso de estas imágenes de contenedores con Docker.

Imágenes de contenedores en Red Hat Container Catalog (RHCC)

Las aplicaciones de misión crítica requieren contenedores de confianza. *Red Hat Container Catalog* es un repositorio de imágenes de contenedores fiables, probadas, certificadas y protegidas basadas en versiones de Red Hat Enterprise Linux (RHEL) y sistemas relacionados. Las imágenes de contenedores disponibles a través de RHCC han experimentado un proceso de garantía de calidad. Todos los componentes fueron rediseñados por Red Hat para evitar vulnerabilidades de seguridad conocidas. Se actualizan regularmente para que contengan la versión requerida de software, incluso cuando no hay una nueva imagen disponible. Con RHCC, puede examinar y buscar imágenes, así como acceder a información acerca de cada imagen, como su versión, contenido y uso.

Búsqueda de imágenes con Quay.io

Quay.io es un repositorio de contenedores avanzado de CoreOS optimizado para la colaboración en equipo. Puede buscar imágenes de contenedores con <https://quay.io/search>.

Al hacer clic en el nombre de una imagen, obtiene acceso a la página de información de la imagen, incluido el acceso a todas las etiquetas existentes para la imagen y el comando a fin de extraer la imagen.

Búsqueda de Dockerfiles en Docker Hub

Cualquier persona puede crear una cuenta de Docker Hub y publicar imágenes de contenedores allí. No hay garantías generales acerca de la calidad y la seguridad; las imágenes en Docker Hub varían desde experimentos soportados profesionalmente a experimentos de una sola vez. Cada imagen debe evaluarse en forma individual.

Después de buscar una imagen, la página de documentación puede proporcionar un enlace a su Dockerfile. Por ejemplo, el primer resultado al buscar **mysql** es la página de documentación para la imagen **MySQL official** en https://hub.docker.com/_/mysql.

En esa página, el enlace para la imagen **5.6/Dockerfile** apunta al proyecto GitHub **docker-library**, que aloja **Dockerfiles** para imágenes compiladas por el sistema de compilación automática de la comunidad de Docker.

La URL directa para el árbol de Docker Hub MySQL 5.6 **Dockerfile** es <https://github.com/docker-library/mysql/blob/master/5.6>.

Descripción del uso de la herramienta de Fuente a imagen (Source-to-Image) de OpenShift

Source-to-Image (S2I) proporciona una alternativa para el uso de Dockerfiles para crear nuevas imágenes de contenedores y puede usarse como una característica para OpenShift o como la utilidad de **s2i** independiente. S2I permite a los desarrolladores trabajar con sus herramientas habituales, en lugar de aprender la sintaxis de Dockerfile y usar los comandos del sistema operativo, como **yum**, y generalmente crea imágenes más ligeras, con menos capas.

S2I usa el siguiente proceso para compilar una imagen de contenedor personalizada para una aplicación:

1. Inicie un contenedor a partir de una imagen de contenedor base denominada *imagen de compilador*, que incluye un tiempo de ejecución del lenguaje de programación y herramientas de desarrollo esenciales, como compiladores y administradores de paquetes.

2. Obtenga el código fuente de la aplicación, generalmente, de un servidor Git, y envíelo al contenedor.
3. Compile los archivos binarios de la aplicación dentro del contenedor.
4. Guarde el contenedor, después de alguna limpieza, como una nueva imagen de contenedor, que incluye el tiempo de ejecución del lenguaje de programación y los binarios de la aplicación.

La imagen del compilador es una imagen de contenedor regular que sigue una estructura de directorio estándar y proporciona scripts que se invocan durante el proceso S2I. La mayoría de estas imágenes del compilador también pueden usarse como imágenes base para Dockerfiles, fuera del proceso S2I.

El comando **s2i** se usa para ejecutar el proceso S2I fuera de OpenShift, en un entorno solo de Docker. Puede instalarse en un sistema RHEL desde el paquete de RPM *fuente a imagen (source-to-image)*, y en otras plataformas, incluidas Windows y Mac OS, desde los instaladores disponibles en el proyecto de S2I en GitHub.



Referencias

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 de RHSCl en GitHub

<https://github.com/sclorg?q=-container>

Uso de imágenes de contenedores de Red Hat Software Collections

<https://access.redhat.com/articles/1752723>

Quay.io

<https://quay.io/search>

Docker Hub

<https://hub.docker.com/>

Proyecto GitHub Docker Library

<https://github.com/docker-library>

Proyecto GitHub S2I

<https://github.com/openshift/source-to-image>

► Cuestionario

Enfoques para el diseño de imágenes de contenedores

Elija las respuestas correctas para las siguientes preguntas:

► 1. ¿Qué método de creación de imágenes de contenedores recomienda la comunidad de contenedores? (Elija una opción).

- a. Ejecutar comandos dentro de un contenedor de sistemas operativos básicos, confirmar el contenedor y guardarla o exportarlo como una nueva imagen de contenedor.
- b. Ejecutar comandos desde Dockerfile y enviar la imagen de contenedor generada a un registro de imágenes.
- c. Crear las capas de imágenes de contenedores de forma manual a partir de archivos tar.
- d. Ejecutar el comando **podman build** para procesar la descripción de imágenes de contenedores en formato YAML.

► 2. ¿Cuales son las dos ventajas de usar el proceso S2I independiente como alternativa a Dockerfiles? (Elija dos opciones).

- a. No requiere herramientas adicionales aparte de una configuración básica de Podman.
- b. Crea imágenes de contenedores más pequeños, que tienen menos capas.
- c. Reutiliza imágenes del compilador de alta calidad.
- d. Actualiza automáticamente la imagen secundaria a medida que se modifica la imagen principal (por ejemplo, con correcciones de seguridad).
- e. Crea imágenes compatibles con OpenShift, a diferencia de las imágenes de contenedores creadas desde herramientas de Docker.

► 3. ¿Cuáles son los dos escenarios típicos para crear un Dockerfile a fin de compilar una imagen secundaria a partir de una imagen existente? (Elija dos opciones).

- a. Agregar nuevas librerías de tiempo de ejecución.
- b. Definir restricciones en el acceso de un contenedor en la CPU de la máquina host.
- c. Agregar librerías internas, que se compartirán como una sola capa de imagen entre varias imágenes de contenedores para diferentes aplicaciones.

► Solución

Enfoques para el diseño de imágenes de contenedores

Elija las respuestas correctas para las siguientes preguntas:

► 1. **¿Qué método de creación de imágenes de contenedores recomienda la comunidad de contenedores? (Elija una opción).**

- a. Ejecutar comandos dentro de un contenedor de sistemas operativos básicos, confirmar el contenedor y guardarlo o exportarlo como una nueva imagen de contenedor.
- b. Ejecutar comandos desde Dockerfile y enviar la imagen de contenedor generada a un registro de imágenes.
- c. Crear las capas de imágenes de contenedores de forma manual a partir de archivos tar.
- d. Ejecutar el comando `podman build` para procesar la descripción de imágenes de contenedores en formato YAML.

► 2. **¿Cuales son las dos ventajas de usar el proceso S2I independiente como alternativa a Dockerfiles? (Elija dos opciones).**

- a. No requiere herramientas adicionales aparte de una configuración básica de Podman.
- b. Crea imágenes de contenedores más pequeños, que tienen menos capas.
- c. Reutiliza imágenes del compilador de alta calidad.
- d. Actualiza automáticamente la imagen secundaria a medida que se modifica la imagen principal (por ejemplo, con correcciones de seguridad).
- e. Crea imágenes compatibles con OpenShift, a diferencia de las imágenes de contenedores creadas desde herramientas de Docker.

► 3. **¿Cuáles son los dos escenarios típicos para crear un Dockerfile a fin de compilar una imagen secundaria a partir de una imagen existente? (Elija dos opciones).**

- a. Agregar nuevas librerías de tiempo de ejecución.
- b. Definir restricciones en el acceso de un contenedor en la CPU de la máquina host.
- c. Agregar librerías internas, que se compartirán como una sola capa de imagen entre varias imágenes de contenedores para diferentes aplicaciones.

Compilación de imágenes de contenedores personalizadas con Dockerfiles

Objetivos

Después de completar esta sección, los estudiantes deberían poder crear una imagen de contenedor con comandos comunes de Dockerfile.

Compilación de contenedores básicos

Un **Dockerfile** es un mecanismo para automatizar la compilación de imágenes de contenedores. La compilación de una imagen a partir de un Dockerfile es un proceso de tres pasos:

1. Crear un directorio de trabajo
2. Escribir el **Dockerfile**
3. Compilar la imagen con Podman

Crear un directorio de trabajo

El *directorio de trabajo* es el directorio que contiene todos los archivos necesarios para compilar la imagen. La creación de un directorio de trabajo vacío es una buena práctica para evitar incorporar archivos innecesarios a la imagen. Por motivos de seguridad, el directorio root, `/`, nunca debería usarse como directorio de trabajo para compilaciones de imágenes.

Escribir la especificación de Dockerfile

Un **Dockerfile** es un archivo de texto que debe existir en el directorio de trabajo. Este archivo contiene las instrucciones necesarias para compilar la imagen. La sintaxis básica de un **Dockerfile** es:

```
# Comment  
INSTRUCTION arguments
```

Las líneas que comienzan con numeral (#) son comentarios. *INSTRUCCIÓN* indica cualquier palabra clave de instrucción de Dockerfile. Las instrucciones no distinguen entre mayúsculas y minúsculas, pero la convención es realizar instrucciones completamente en mayúscula para mejorar la visibilidad.

La primera instrucción que no sea un comentario debe ser una instrucción **FROM** para especificar la imagen base. Las instrucciones de Dockerfile se ejecutan en un nuevo contenedor usando esta imagen y, luego, se confirman en una nueva imagen. La siguiente instrucción (si hubiera) se ejecuta en esa nueva imagen. El orden de ejecución de las instrucciones es el orden de su aparición en el Dockerfile.



nota

La instrucción **ARG** puede aparecer antes de la instrucción **FROM**, pero las instrucciones **ARG** están fuera de los objetivos para esta sección.

Cada instrucción Dockerfile se ejecuta en un contenedor independiente usando una imagen intermedia compilada de todos los comandos anteriores. Esto significa que cada instrucción es independiente de otras instrucciones en el Dockerfile.

A continuación, se incluye un ejemplo de Dockerfile para compilar un contenedor de servidores web simple de Apache:

```
# This is a comment line ①
FROM ubi7/ubi:7.7 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 Las líneas que comienzan con numeral (#) son comentarios.
- ② La instrucción **FROM** declara que la nueva imagen de contenedor se extiende a la imagen base del contenedor **ubi7/ubi:7.7**. Dockerfiles puede usar cualquier otra imagen de contenedor como imagen base, no solo imágenes de distribuciones de sistemas operativos. Red Hat proporciona un conjunto de imágenes de contenedor que están certificadas y probadas, y recomienda encarecidamente el uso de estas imágenes de contenedor como base.
- ③ **LABEL** es responsable de agregar metadatos genéricos a una imagen. Una instrucción **LABEL** es un par de clave-valor simple.
- ④ **MAINTAINER** indica el campo **Author** (Autor) de los metadatos de la imagen del contenedor generada. Puede usar el comando **podman inspect** para ver los metadatos de la imagen.
- ⑤ **RUN** ejecuta comandos en una nueva capa sobre la imagen actual. La shell que se usa para ejecutar comandos es **/bin/sh**.
- ⑥ **EXPOSE** indica que el contenedor escucha en el puerto de red especificado en tiempo de ejecución. La instrucción **EXPOSE** define metadatos únicamente; no permite que se pueda acceder a los puertos desde el host. La opción **-p** en el comando **podman run** expone puertos de contenedor desde el host.
- ⑦ **ENV** es responsable de definir las variables de entorno que están disponibles en el contenedor. Puede declarar varias instrucciones **ENV** dentro de **Dockerfile**. Puede usar el comando **env** dentro del contenedor para ver cada una de las variables de entorno.
- ⑧ La instrucción **ADD** copia archivos o carpetas de una fuente local o remota y los agrega al sistema de archivos del contenedor. Si se usa para copiar archivos locales, estos deben estar en el directorio de trabajo. La instrucción **ADD** desempaquetla los archivos de **.tar** locales en el directorio de la imagen de destino.
- ⑨ **COPY** copia archivos desde el directorio de trabajo y los agrega al sistema de archivos del contenedor. No es posible copiar un archivo remoto mediante su URL con esta instrucción de Dockerfile.
- ⑩ **USER** especifica el nombre de usuario o el UID que se usará cuando se ejecute la imagen de contenedor para las instrucciones **RUN**, **CMD** y **ENTRYPOINT**. La práctica adecuada es definir un usuario diferente a **root** por motivos de seguridad.
- ⑪ **ENTRYPOINT** especifica el comando predeterminado para ejecutarse cuando se ejecuta la imagen en un contenedor. Si se omite, el valor predeterminado de **ENTRYPOINT** es **/bin/sh -c**.

- ⑫ **CMD** proporciona los argumentos predeterminados para la instrucción **ENTRYPOINT**. Si aplica el valor de **ENTRYPOINT** predeterminado (**/bin/sh -c**), **CMD** forma un comando ejecutable y los parámetros que se ejecutan al inicio del contenedor.

CMD y ENTRYPOINT

Las instrucciones **ENTRYPOINT** y **CMD** tienen dos formatos:

- Forma ejec. (uso de un arreglo JSON):

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

- Forma de shell:

```
ENTRYPOINT command param1 param2
CMD param1 param2
```

La forma ejec. es la forma preferida. La forma de shell envuelve los comandos en una shell de **/bin/sh -c** y crea un proceso de shell a veces innecesario. Además, algunas combinaciones no están permitidas o pueden no funcionar como se espera. Por ejemplo, si **ENTRYPOINT** es **["ping"]** (forma ejec.) y **CMD** es **localhost** (forma shell), el comando ejecutado esperado es **ping localhost**, pero el contenedor prueba **ping /bin/sh -c localhost**, que es un comando dañado.

Dockerfile debería contener, como máximo, una instrucción **ENTRYPOINT** y una instrucción **CMD**. Si hay más de una de cada una, solo la última instrucción tendrá efecto. **CMD** puede estar presente sin especificar un **ENTRYPOINT**. En este caso, se aplica el **ENTRYPOINT** de la imagen base o el **ENTRYPOINT** predeterminado si no se define ninguno.

Podman puede anular la instrucción **CMD** cuando se inicia un contenedor. Si están presentes, todos los parámetros para el comando **podman run** después del nombre de la imagen forman la instrucción **CMD**. Por ejemplo, la siguiente instrucción hace que el contenedor que se está ejecutando muestre la hora actual:

```
ENTRYPOINT ["/bin/date", "+%H:%M"]
```

El **ENTRYPOINT** define tanto el comando que se ejecutará como los parámetros. Entonces, no se puede usar la instrucción **CMD**. En el siguiente ejemplo, se proporciona la misma funcionalidad, con el beneficio agregado de que se puede sobrescribir la instrucción **CMD** cuando se inicia un contenedor:

```
ENTRYPOINT ["/bin/date"]
CMD ["+%H:%M"]
```

En ambos casos, cuando se inicia un contenedor sin proporcionar un parámetro, se muestra la hora actual:

```
[student@workstation ~]$ sudo podman run -it do180/rhel
11:41
```

capítulo 5 | Creación de imágenes de contenedores personalizadas

En el segundo caso, si aparece un parámetro después del nombre de una imagen en el comando **podman run**, sobrescribe la instrucción **CMD**. El siguiente comando muestra el día actual de la semana en lugar de la hora:

```
[student@workstation demo-basic]$ sudo podman run -it do180/rhel +%A  
Tuesday
```

Otro enfoque es usar el **ENTRYPOINT** predeterminado y la instrucción **CMD** para definir el comando inicial. La siguiente instrucción muestra la hora actual, con el beneficio agregado de poder anularse en tiempo de ejecución.

```
CMD ["date", "+%H:%M"]
```

ADD y COPY

Las instrucciones **ADD** y **COPY** tienen dos formas:

- Forma de shell:

```
ADD <source>... <destination>  
COPY <source>... <destination>
```

- Forma ejec.:

```
ADD ["<source>", ... "<destination>"]  
COPY ["<source>", ... "<destination>"]
```

Si el *origen* es una ruta del sistema de archivos, debe estar dentro del directorio de trabajo.

La instrucción **ADD** también le permite especificar un recurso con una URL:

```
ADD http://someserver.com/filename.pdf /var/www/html
```

Si el *origen* es un archivo comprimido, la instrucción **ADD** descomprime el archivo en la carpeta de *destino*. La instrucción **COPY** no tiene esta funcionalidad.



Advertencia

Las instrucciones **ADD** y **COPY** copian los archivos, conservando los permisos, con **root** como propietario, aun si se especifica la instrucción **USER**. Red Hat recomienda el uso de una instrucción **RUN** después de la copia para cambiar el propietario y evitar errores de "permiso denegado".

Imagen en capas

Cada instrucción en un **Dockerfile** crea una nueva capa de imagen. Tener demasiadas instrucciones en un **Dockerfile** crea demasiadas capas, lo que genera imágenes más grandes. Por ejemplo, considere la siguiente instrucción **RUN** en un **Dockerfile**:

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms"  
RUN yum update -y  
RUN yum install -y httpd
```

El ejemplo anterior no es una práctica adecuada cuando se crean imágenes de contenedores. Crea tres capas (una para cada instrucción **RUN**) mientras que solo la última es significativa. Red Hat recomienda minimizar la cantidad de capas. También puede lograr el mismo objetivo al crear una sola capa si usa la conjunción **&&**:

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms" && yum update -y && yum  
install -y httpd
```

El problema con este enfoque es que se deteriora la legibilidad de **Dockerfile**. Use el código de escape \ para insertar saltos de línea y mejorar la legibilidad. También puede aplicar sangría a las líneas para alinear los comandos:

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms" && \  
    yum update -y && \  
    yum install -y httpd
```

Este ejemplo crea solo una capa y mejora la legibilidad. Las instrucciones **RUN**, **COPY** y **ADD** crean nuevas capas de imágenes, pero **RUN** puede mejorarse de este modo.

Red Hat recomienda la aplicación de reglas de formato similares a otras instrucciones que aceptan varios parámetros, como **LABEL** y **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"
```

Compilación de imágenes con Podman

El comando **podman build** procesa el **Dockerfile** y compila una nueva imagen sobre la base de las instrucciones que contiene. La sintaxis para este comando es la siguiente:

```
$ podman build -t NAME:TAG DIR
```

DIR es la ruta al directorio de trabajo, que debe incluir el **Dockerfile**. Puede ser el directorio actual, según se designe con un punto (.), si el directorio de trabajo es el directorio actual.

NAME:TAG es el nombre con una etiqueta que se asigna a la nueva imagen. Si *TAG* no se especifica, la imagen se etiqueta automáticamente como **latest** (más reciente).



Referencias

Guía de referencia de Dockerfile

<https://docs.docker.com/engine/reference/builder/>

Creación de imágenes base

<https://docs.docker.com/engine/userguide/eng-image/baseimages/>

► Ejercicio Guiado

Creación de una imagen de contenedor de Apache básica

En este ejercicio, creará una imagen de contenedor de Apache básica.

Resultados

Deberá ser capaz de crear una imagen de contenedor de Apache personalizada creada sobre una imagen de Red Hat Enterprise Linux 7.5.

Andes De Comenzar

Ejecute el siguiente comando para descargar los archivos relevantes del trabajo de laboratorio y para verificar que se esté ejecutando Docker:

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

► 1. Cree el Dockerfile de Apache.

- 1.1. Abra un terminal en **workstation**. Use su editor preferido y cree un nuevo Dockerfile:

```
[student@workstation ~]$ vim /home/student/D0180/labs/dockerfile-create/Dockerfile
```

- 1.2. Use UBI 7.7 como imagen base mediante la adición de la siguiente instrucción **FROM** en la parte superior del nuevo Dockerfile:

```
FROM ubi7/ubi:7.7
```

- 1.3. Debajo de la instrucción **FROM**, incluya la instrucción **MAINTAINER** para configurar el campo **Author** (Autor) en la nueva imagen. Reemplace los valores para incluir su nombre y su dirección de correo electrónico:

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

- 1.4. Debajo de la instrucción **MAINTAINER**, agregue la siguiente instrucción **LABEL** para agregar metadatos de descripción a la nueva imagen:

```
LABEL description="A custom Apache container based on UBI 7"
```

- 1.5. Agregue una instrucción **RUN** con un comando **yum install** para instalar Apache en el nuevo contenedor:

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

16. Agregue una instrucción **RUN** para reemplazar el contenido de la página de inicio HTTPD predeterminada:

```
RUN echo "Hello from Dockerfile" > /usr/share/httpd/noindex/index.html
```

17. Use la instrucción **EXPOSE** debajo de la instrucción **RUN** para documentar el puerto que escucha el contenedor en tiempo de ejecución. En esta instancia, establezca el puerto en 80, ya que es el puerto predeterminado para un servidor de Apache:

```
EXPOSE 80
```

**nota**

La instrucción **EXPOSE** no pone realmente el puerto especificado a disposición del host; en su lugar, la instrucción sirve como metadatos acerca de los puertos en los que escucha el contenedor.

18. Al final del archivo, use la siguiente instrucción **ENTRYPOINT** para establecer **httpd** como punto de entrada predeterminado:

```
ENTRYPOINT ["httpd", "-D", "FOREGROUND"]
```

19. Verifique que su Dockerfile coincida con lo siguiente antes de guardar y continuar con los siguientes pasos:

```
FROM ubi7/ubi:7.7

MAINTAINER Your Name <youremail>

LABEL description="A basic Apache container on RHEL 7 UBI"

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

RUN echo "Hello from Dockerfile" > /usr/share/httpd/noindex/index.html

EXPOSE 80

ENTRYPOINT ["httpd", "-D", "FOREGROUND"]
```

► 2. Compile y verifique la imagen de contenedor de Apache.

21. Use los siguientes comandos para crear una imagen de contenedor de Apache básica con el Dockerfile creado recientemente:

```
[student@workstation ~]$ cd /home/student/D0180/labs/dockerfile-create
[student@workstation dockerfile-create]$ sudo podman build --layers=false \
> -t do180/apache .
STEP 1: FROM ubi7/ubi:7.7
Getting image source signatures ①
Copying blob sha256:...output omitted...
71.46 MB / 71.46 MB [=====] 18s
```

```
...output omitted...
Storing signatures
STEP 2: MAINTAINER username <username@example.com>
STEP 3: LABEL description="A basic Apache container on RHEL 7"
STEP 4: RUN yum install -y httpd &&      yum clean all
Loaded plugins: ovl, product-id, search-disabled-repos, subscription-manager
...output omitted...
Complete!
STEP 5: RUN echo "Hello from Dockerfile" > /usr/share/httpd/noindex/index.html
STEP 6: EXPOSE 80
STEP 7: ENTRYPOINT ["httpd", "-D", "FOREGROUND"]
ERRO[0109] HOSTNAME is not supported for OCI image format...output omitted...②
STEP 8: COMMIT ...output omitted... localhost/do180/apache:latest
Getting image source signatures
...output omitted...
Storing signatures
--> 190a...95c5
```

- ① La imagen de contenedor enumerada en la instrucción **FROM** solo se descarga si no está ya presente en el almacenamiento local.
- ② El error es benigno y se puede omitir. Es un problema conocido de Podman (consulte Error 1634806 [https://bugzilla.redhat.com/show_bug.cgi?id=1634806]) y desaparecerá en versiones posteriores.

- 2.2. Despues de que haya finalizado el proceso de compilación, ejecute **podman images** para ver la nueva imagen en el repositorio de imágenes:

```
[student@workstation dockerfile-create]$ sudo podman images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
localhost/do180/apache    latest   c69affe9d93b  33 seconds ago  247MB
registry.access.redhat.com/ubi7/ubi latest   6fecccc91c83  4 weeks ago   215MB
```



nota

Podman crea muchas imágenes intermedias anónimas durante el proceso de compilación. No se muestran, a menos que se use **-a**. Use la opción **--layers=false** del subcomando **build** para indicar a Podman que elimine imágenes intermedias.

- 3. Ejecute el contenedor de Apache.

- 3.1. Use el siguiente comando para ejecutar un contenedor con la imagen de Apache:

```
[student@workstation dockerfile-create]$ sudo podman run --name lab-apache \
> -d -p 10080:80 do180/apache
fa1d1c450e8892ae085dd8bbf763edac92c41e6ffaa7ad6ec6388466809bb391
```

- 3.2. Ejecute el comando **podman ps** para ver el contenedor que se está ejecutando:

```
[student@workstation dockerfile-create]$ sudo podman ps
CONTAINER ID IMAGE                  COMMAND            ...output omitted...
fa1d1c450e88 localhost/do180/apache:latest httpd -D FOREGROU...output omitted...
```

- 3.3. Use el comando **curl** para verificar que el servidor se esté ejecutando:

```
[student@workstation dockerfile-create]$ curl 127.0.0.1:10080
Hello from Dockerfile
```

Finalizar

En **workstation**, ejecute el script **lab dockerfile-create finish** para terminar este trabajo de laboratorio.

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

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Creación de imágenes de contenedores personalizadas

Lista de verificación de rendimiento

En este trabajo de laboratorio, creará un Dockerfile para compilar una imagen de contenedor personalizada del servidor web Apache. La imagen personalizada se basará en una imagen de UBI 7.7 de RHEL y servirá a una página `index.html` personalizada.

Resultados

Debería poder crear un contenedor del servidor web Apache personalizado que aloje archivos HTML estáticos.

Andes De Comenzar

Abra un terminal en `workstation` como usuario `student` y ejecute el siguiente comando:

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

- Revise el stub de Dockerfile provisto en la carpeta `/home/student/D0180/labs/dockerfile-review/`. Edite el `Dockerfile` y asegúrese de que cumpla con las siguientes especificaciones:
 - La imagen base es `ubi7/ubi:7.7`
 - Define el nombre de autor deseado y el ID de correo electrónico con la instrucción `MAINTAINER`.
 - Establece la variable de entorno: `PORT` en 8080
 - Instale Apache (paquete `httpd`).
 - Cambie el archivo de configuración de Apache `/etc/httpd/conf/httpd.conf` para escuchar el puerto 8080 en lugar del puerto predeterminado 80.
 - Cambie la titularidad de las carpetas `/etc/httpd/logs` y `/run/httpd` para el usuario y el grupo `apache` (UID y GID = 48).
 - Para que los usuarios del contenedor sepan cómo acceder al servidor web Apache, exponga el valor establecido en la variables de entorno `PORT`.
 - Copie el contenido de la carpeta `src/` en el directorio del trabajo de laboratorio en el archivo `DocumentRoot` de Apache (`/var/www/html/`) dentro del contenedor.
 - La carpeta `src` contiene un solo archivo `index.html` que imprime un mensaje `Hello World!` (Hola, mundo).
 - Inicie el daemon de `httpd` de Apache en primer plano con el siguiente comando:

```
httpd -D FOREGROUND
```

2. Compile la imagen personalizada de Apache con el nombre **do180/custom-apache**.
3. Cree un nuevo contenedor en modo separado con las siguientes características:
 - Nombre: **dockerfile**
 - Imagen de contenedor: **do180/custom-apache**
 - Reenvío de puertos: desde el puerto 20080 del host hasta el puerto 8080 del contenedor
 - Ejecutar como daemon: síVerifique que el contenedor esté listo y ejecutándose.
4. Verifique que el servidor sirva al archivo HTML.

Evaluación

Evalúe su trabajo ejecutando el comando **lab dockerfile-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el comando **lab dockerfile-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

► Solución

Creación de imágenes de contenedores personalizadas

Lista de verificación de rendimiento

En este trabajo de laboratorio, creará un Dockerfile para compilar una imagen de contenedor personalizada del servidor web Apache. La imagen personalizada se basará en una imagen de UBI 7.7 de RHEL y servirá a una página `index.html` personalizada.

Resultados

Debería poder crear un contenedor del servidor web Apache personalizado que aloje archivos HTML estáticos.

Andes De Comenzar

Abra un terminal en `workstation` como usuario `student` y ejecute el siguiente comando:

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

- Revise el stub de Dockerfile provisto en la carpeta `/home/student/D0180/labs/dockerfile-review/`. Edite el `Dockerfile` y asegúrese de que cumpla con las siguientes especificaciones:
 - La imagen base es `ubi7/ubi:7.7`
 - Define el nombre de autor deseado y el ID de correo electrónico con la instrucción `MAINTAINER`.
 - Establece la variable de entorno: `PORT` en 8080
 - Instale Apache (paquete `httpd`).
 - Cambie el archivo de configuración de Apache `/etc/httpd/conf/httpd.conf` para escuchar el puerto 8080 en lugar del puerto predeterminado 80.
 - Cambie la titularidad de las carpetas `/etc/httpd/logs` y `/run/httpd` para el usuario y el grupo `apache` (UID y GID = 48).
 - Para que los usuarios del contenedor sepan cómo acceder al servidor web Apache, exponga el valor establecido en la variables de entorno `PORT`.
 - Copie el contenido de la carpeta `src/` en el directorio del trabajo de laboratorio en el archivo `DocumentRoot` de Apache (`/var/www/html/`) dentro del contenedor.
 - La carpeta `src` contiene un solo archivo `index.html` que imprime un mensaje `Hello World!` (Hola, mundo).
 - Inicie el daemon de `httpd` de Apache en primer plano con el siguiente comando:

```
httpd -D FOREGROUND
```

- 1.1. Abra un terminal (**Applications [Aplicaciones]** → **System Tools [Herramientas del sistema]** → **Terminal [Terminal]**) y edite el Dockerfile ubicado en la carpeta **/home/student/D0180/labs/dockerfile-review/**.

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

- 1.2. Establezca la imagen base para el Dockerfile en **ubi7/ubi:7.7**.

```
FROM ubi7/ubi:7.7
```

- 1.3. Establezca su nombre y correo electrónico con una instrucción **MAINTAINER**.

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

- 1.4. Cree una variable de entorno denominada **PORT** y establezcala en 8080.

```
ENV PORT 8080
```

- 1.5. Instale el servidor Apache.

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

- 1.6. Cambie el archivo de configuración del servidor HTTP Apache para escuchar el puerto 8080 y cambiar la titularidad de las carpetas de trabajo del servidor con una sola instrucción **RUN**.

```
RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" /etc/httpd/conf/httpd.conf && \  
    chown -R apache:apache /etc/httpd/logs/ && \  
    chown -R apache:apache /run/httpd/
```

- 1.7. Use la instrucción **USER** para ejecutar el contenedor con el usuario **apache**. Use la instrucción **EXPOSE** para documentar el puerto que escucha el contenedor en tiempo de ejecución. En esta instancia, establezca el puerto en la variable de entorno **PORT**, que es la predeterminada para un servidor Apache.

```
USER apache
```

```
# Expose the custom port that you provided in the ENV var  
EXPOSE ${PORT}
```

- 1.8. Copie todos los archivos de la carpeta **src** en la ruta **DocumentRoot** de Apache en **/var/www/html**.

```
# Copy all files under src/ folder to Apache DocumentRoot (/var/www/html)
COPY ./src/ /var/www/html/
```

- 1.9. Finalmente, inserte una instrucción **CMD** para ejecutar **httpd** en primer plano y, luego, guarde el Dockerfile.

```
# Start Apache in the foreground
CMD ["httpd", "-D", "FOREGROUND"]
```

2. Compile la imagen personalizada de Apache con el nombre **do180/custom-apache**.

- 2.1. Verifique el Dockerfile para la imagen personalizada de Apache.

El Dockerfile para la imagen personalizada de Apache debería verse similar al siguiente:

```
FROM ubi7/ubi:7.7

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. Ejecute un comando **sudo podman build** para compilar la imagen personalizada de Apache y asígnele el nombre **do180/custom-apache**.

```
[student@workstation dockerfile-review]$ sudo podman build \
> -t do180/custom-apache .
STEP 1: FROM ubi7/ubi:7.7
...output omitted...
STEP 2: MAINTAINER username <username@example.com>
...output omitted...
--> 6755...ffc7
STEP 3: FROM 6755...ffc7

STEP 4: ENV PORT 8080
--> 0ff7...e4f0
```

```
STEP 5: FROM 0ff7...e4f0
STEP 6: RUN yum install -y httpd &&      yum clean all
...output omitted...

Installed:
httpd.x86_64 0:2.4.6-88.el7

...output omitted...
--> d398...e874
STEP 13: FROM d398...e874
STEP 14: COPY ./src/ /var/www/html/
--> 7de1...0432
STEP 15: FROM 7de1...0432
STEP 16: CMD ["httpd", "-D", "FOREGROUND"]
--> da92...b3bd
STEP 17: COMMIT do180/custom-apache
```

- 2.3. Ejecute el comando **podman images** para verificar que la imagen personalizada se compiló correctamente.

```
[student@workstation dockerfile-review]$ sudo podman images
REPOSITORY                                     TAG      IMAGE ID      ...
localhost/do180/custom-apache                latest   da92b9426325 ...
registry.access.redhat.com/ubi7/ubi           7.7     6fecccc91c83 ...
```

3. Cree un nuevo contenedor en modo separado con las siguientes características:

- Nombre: **dockerfile**
- Imagen de contenedor:**do180/custom-apache**
- Reenvío de puertos: desde el puerto 20080 del host hasta el puerto 8080 del contenedor
- Ejecutar como daemon: sí

Verifique que el contenedor esté listo y ejecutándose.

- 3.1. Cree y ejecute el contenedor.

```
[student@workstation dockerfile-review]$ sudo podman run -d \
> --name dockerfile -p 20080:8080 do180/custom-apache
367823e35c4a...
```

- 3.2. Verifique que el contenedor esté listo y ejecutándose.

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

4. Verifique que el servidor sirva al archivo HTML.

Ejecute un comando **curl** en 127.0.0.1:20080

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

La salida debería verse de la siguiente manera:

```
<html>
<header><title>DO180 Hello!</title></header>
<body>
  Hello World! The dockerfile-review lab works!
</body>
</html>
```

Evaluación

Evalúe su trabajo ejecutando el comando **lab dockerfile-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el comando **lab dockerfile-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Un **Dockerfile** contiene instrucciones que especifican cómo construir una imagen de contenedor.
- Las imágenes de contenedores provistas por Red Hat Container Catalog o Quay.io son un buen punto de partida para crear imágenes personalizadas para un lenguaje o una tecnología específicos.
- La compilación de una imagen a partir de un Dockerfile es un proceso de tres pasos:
 1. Crear un directorio de trabajo.
 2. Especificar las instrucciones de compilación en un archivo **Dockerfile**.
 3. Compile la imagen con el comando **podman build**.
- El proceso de fuente a imagen (Source-to-Image, S2I) proporciona una alternativa para Dockerfiles. S2I implementa un proceso de compilación de imágenes de contenedores estandarizado para tecnologías comunes a partir del código fuente de la aplicación. Esto permite a los desarrolladores centrarse en el desarrollo de aplicaciones y no en el desarrollo de Dockerfile.

capítulo 6

Implementación de aplicaciones contenedorizadas en OpenShift

Meta

Implementar aplicaciones con un solo contenedor en OpenShift Container Platform.

Objetivos

- Describir la arquitectura de Kubernetes y Red Hat OpenShift Container Platform.
- Crear recursos de Kubernetes estándares.
- Crear una ruta a un servicio.
- Compilar una aplicación con la utilidad de fuente a imagen (source-to-image) de OpenShift Container Platform.
- Crear una aplicación con la consola web de OpenShift.

Secciones

- Descripción de la arquitectura de Kubernetes y OpenShift (y cuestionario)
- Creación de recursos de Kubernetes (y ejercicio guiado)
- Creación de rutas (y ejercicio guiado)
- Creación de aplicaciones con la utilidad de fuente a imagen (source-to-image) (y ejercicio guiado)
- Creación de aplicaciones con la consola web de OpenShift (y ejercicio guiado)

Trabajo de laboratorio

- Implementación de aplicaciones en contenedores en OpenShift

Descripción de la arquitectura de Kubernetes y OpenShift

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Describir la arquitectura de un clúster de Kubernetes que se ejecuta en Red Hat OpenShift Container Platform (RHOC).
- Enumerar los principales tipos de recursos provistos por Kubernetes y RHOC.
- Identificar las características de red de contenedores, Kubernetes y RHOC.
- Enumerar los mecanismos para que se pueda acceder a un pod de forma externa.

Kubernetes y OpenShift

En capítulos anteriores, vimos que Kubernetes es un servicio de orquestación que simplifica la implementación, la administración y el escalamiento de las aplicaciones contenerizadas. Una de las principales ventajas de usar Kubernetes es que usa varios nodos para garantizar la flexibilidad y la escalabilidad de sus aplicaciones administradas. Kubernetes forma un clúster de servidores de nodos que ejecutan contenedores y se administran de forma central mediante un conjunto de servidores maestros. Un servidor puede actuar como servidor y como nodo, pero estos roles están generalmente separados para brindar mayor estabilidad.

Terminología de Kubernetes

Término	Definición
Nodo	Un servidor que aloja aplicaciones en un clúster de Kubernetes.
Nodo maestro	Un servidor de nodo que administra el plano de control en un clúster de Kubernetes. Los nodos maestros proporcionan servicios de clúster básicos, como API o controladores.
Nodo de trabajador	También llamados nodos de cómputo , los nodos de trabajador ejecutan cargas de trabajo para el clúster. Los pods de aplicación están programados en nodos de trabajador.
Recurso	Los recursos son cualquier tipo de definición de componentes administrada por Kubernetes. Los recursos contienen la configuración del componente administrado (por ejemplo, el rol asignado a un nodo) y el estado actual del componente (por ejemplo, si el nodo está disponible).
Controlador	Un controlador es un proceso de Kubernetes que vigila los recursos y realiza cambios con el fin de intentar mover el estado actual hacia el estado deseado.
Etiqueta	Un par de clave-valor que puede asignarse a un recurso de Kubernetes. Los selectores usan etiquetas con el fin de filtrar recursos elegibles para programaciones y otras operaciones.

Término	Definición
Espacio de nombres	Un alcance para los recursos y procesos de Kubernetes, de modo que los recursos con el mismo nombre se puedan usar en diferentes límites.

**nota**

Las últimas versiones de Kubernetes implementan muchos controladores como **operadores**. Los operadores son componentes de complementos (plug-ins) de Kubernetes que pueden reaccionar ante eventos de clúster y controlar el estado de los recursos. Los operadores y CoreOS Operator Framework están fuera del alcance de este documento.

Red Hat OpenShift Container Platform es un conjunto de componentes y servicios modulares desarrollados sobre la base de Red Hat CoreOS y Kubernetes. RHOCP agrega capacidades de PaaS, como administración remota, mayor seguridad, monitoreo y auditoría, administración del ciclo de vida de la aplicación e interfaces de autoservicio para desarrolladores.

Un clúster de OpenShift es un clúster de Kubernetes que se puede administrar de la misma forma, aunque usando las herramientas de administración provistas por OpenShift, como la interfaz de línea de comandos o la consola web. Esto permite flujos de trabajo más productivos y facilita mucho las tareas comunes.

Terminología de OpenShift

Término	Definición
Infra Nodo	Servidor de nodo que contiene servicios de infraestructura, como monitoreo, registro o enrutamiento externo.
Consola	Interfaz de usuario web proporcionada por el clúster RHOCP que permite a los desarrolladores y administradores interactuar con los recursos del clúster.
Proyecto (Project)	Extensión de OpenShift de los espacios de nombres de Kubernetes. Permite la definición del control de acceso de usuario (UAC) para los recursos.

En el siguiente esquema, se ilustra la pila (stack) de OpenShift Container Platform.

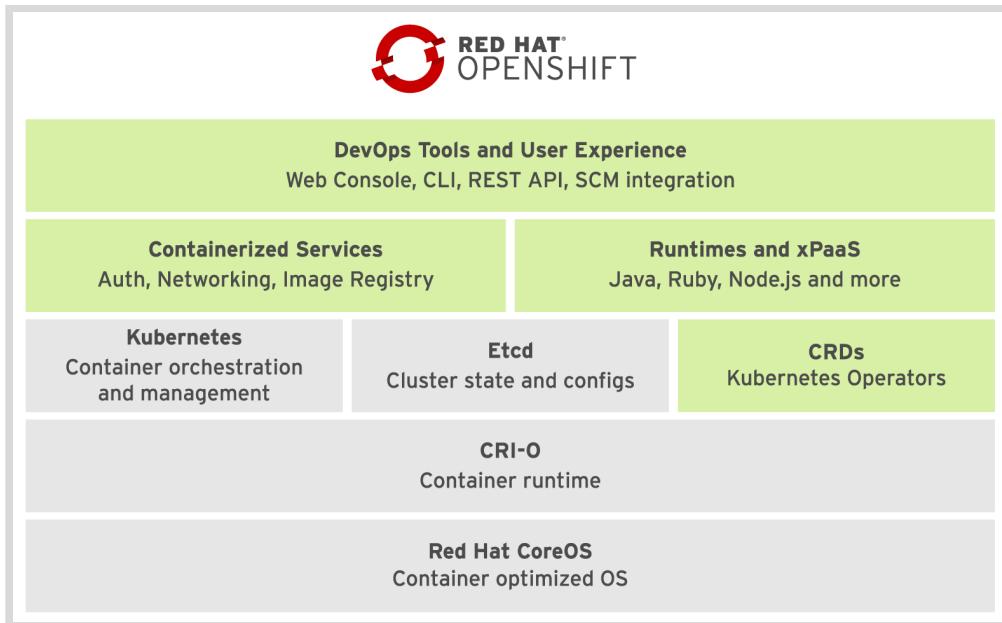


Figura 6.1: Pila (stack) de componentes OpenShift

Desde abajo hacia arriba y de izquierda a derecha, se muestra la infraestructura de contenedores básica, integrada y mejorada por Red Hat:

- El sistema operativo base es Red Hat CoreOS. Red Hat CoreOS es una distribución de Linux enfocada en proporcionar un sistema operativo immutable para la ejecución de contenedores.
- CRI-O es una implementación del Kubernetes CRI (Container Runtime Interface, interfaz de tiempo de ejecución del contenedor) para habilitar el uso de tiempos de ejecución compatibles con OCI (Open Container Initiative, iniciativa de contenedor abierto). CRI-O puede usar el tiempo de ejecución de cualquier contenedor que satisfaga la CRI: **runc** (usado por el servicio Docker), **libpod** (usado por Podman) o **rkt** (desde CoreOS).
- Kubernetes administra un clúster de hosts (físicos o virtuales) que ejecutan contenedores. Usa recursos que describen aplicaciones con varios contenedores compuestos por varios recursos y el modo en que se interconectan.
- *Etc* es un almacén de claves-valores distribuido, usado por Kubernetes para almacenar información de estado y configuración acerca de los contenedores y otros recursos dentro del clúster de Kubernetes.
- Las *definiciones de recursos personalizados (CRD)* son tipos de recursos almacenados en Etc y administrados por Kubernetes. Estos tipos de recursos forman el estado y la configuración de todos los recursos administrados por OpenShift.
- Los servicios en contenedores cumplen muchas funciones de infraestructura de PaaS, como redes y autorización. RHOPC usa la infraestructura de contenedores básica de Kubernetes y el tiempo de ejecución de contenedores subyacentes para la mayoría de las funciones internas. Es decir, la mayoría de los servicios internos de RHOPC se ejecutan como contenedores organizados por Kubernetes.
- Tiempos de ejecución y xPaaS: son imágenes de contenedores base listas para ser usadas por desarrolladores; cada una está configurada previamente con una base de datos o un lenguaje de tiempo de ejecución en particular. La oferta xPaaS es un conjunto de imágenes base para productos de middleware Red Hat, como JBoss EAP y ActiveMQ. *Red Hat OpenShift Application Runtimes (RHOAR)* es un conjunto de tiempos de ejecución optimizados para

aplicaciones nativas en la nube en OpenShift. Los tiempos de ejecución de las aplicaciones disponibles son Red Hat JBoss EAP, OpenJDK, Thorntail, Eclipse Vert.x, Spring Boot y Node.js.

- Herramientas de DevOps y experiencia del usuario: RHOCP proporciona herramientas de administración CLI y de interfaz de usuario (UI) web para administrar aplicaciones del usuario y servicios de RHOCP. Las herramientas CLI y UI web de OpenShift están desarrolladas a partir de las API REST, que pueden usar las herramientas externas, como IDE y plataformas de integración continua.

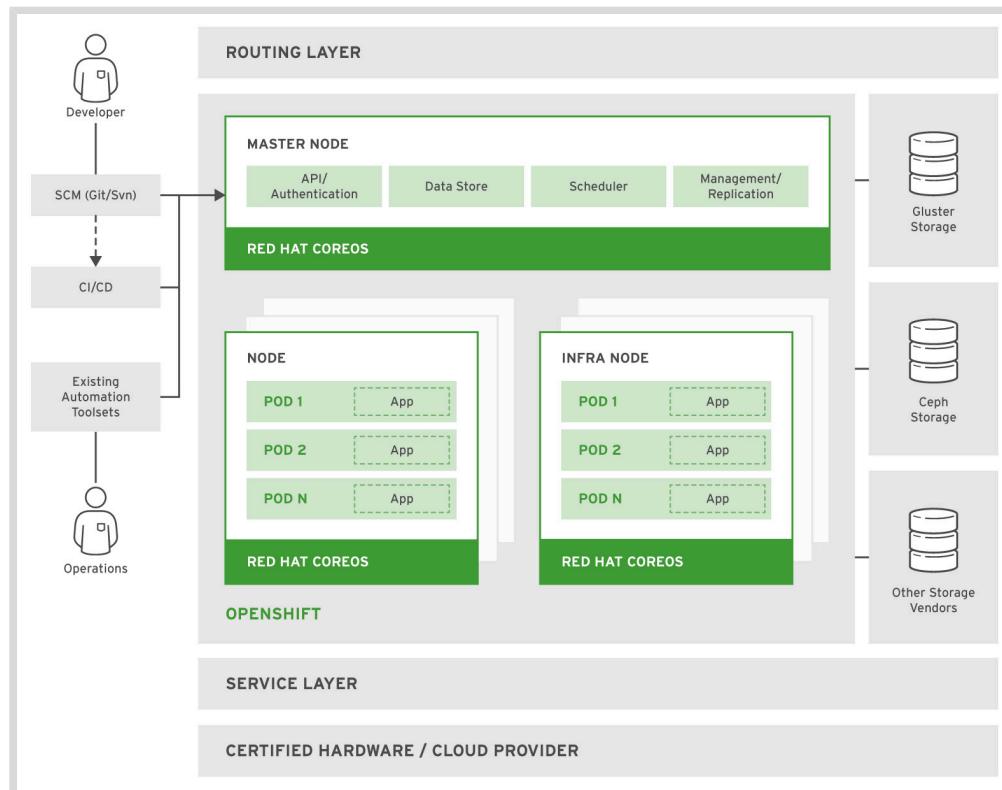


Figura 6.2: Arquitectura de OpenShift y Kubernetes

Nuevas características en RHOCP 4

RHOCP 4 es un cambio significativo de versiones anteriores. Además de mantener la compatibilidad con versiones anteriores, incluye nuevas características, como:

- CoreOS como sistema operativo obligatorio para todos los nodos, que ofrece una infraestructura inmutable optimizada para contenedores.
- Un nuevo instalador de clústeres que guía el proceso de instalación y actualización.
- Una plataforma que se administra de forma automática, capaz de aplicar automáticamente las actualizaciones y recuperaciones de clústeres sin interrupciones.
- Una administración del ciclo de vida de aplicaciones rediseñado.
- Un SDK del operador para compilar, probar y empaquetar operadores.

Descripción de tipos de recursos de Kubernetes

Kubernetes tiene seis tipos de recursos principales que pueden crearse y configurarse con un archivo YAML o JSON, o con herramientas de administración de OpenShift:

Pods (po)

Representan una colección de contenedores que comparten recursos, como direcciones IP y volúmenes de almacenamiento persistente. Es la unidad básica de trabajo para Kubernetes.

Servicios (svc)

Definen una combinación simple de IP/puerto que proporciona acceso a un pool (conjunto) de pods. De forma predeterminada, los servicios conectan clientes con pods de manera Round-Robin.

Controladores de replicación (rc)

Un recurso de Kubernetes que define cómo se replican los pods (escalado horizontalmente) en diferentes nodos. Los controladores de replicación son un servicio básico de Kubernetes para proporcionar una alta disponibilidad para pods y contenedores.

Volúmenes persistentes (pv)

Definen las áreas de almacenamiento que usarán los pods de Kubernetes.

Reclamaciones de volumen persistente (pvc)

Representan una solicitud de almacenamiento por parte de un pod. Las PVC conectan un PV a un pod para que sus contenedores puedan usarlo, generalmente al montar el almacenamiento en el sistema de archivos del contenedor.

ConfigMaps (cm) y Secrets

Contiene un conjunto de claves y valores que pueden ser usados por otros recursos. ConfigMaps y Secrets se usan generalmente para centralizar los valores de configuración usados por varios recursos. Los secretos difieren de los mapas de ConfigMaps en que los valores de Secrets están siempre codificados (no cifrados) y su acceso está restringido a menos usuarios autorizados.

Si bien los pods de Kubernetes pueden crearse en forma independiente, generalmente son creados por recursos de alto nivel, como los controladores de replicación.

Tipos de recursos de OpenShift

A continuación, se detallan los tipos de recursos principales agregados por OpenShift Container Platform a Kubernetes:

Configuración de implementación (dc)

Representa el conjunto de contenedores incluidos en un pod y las estrategias de implementación que se usarán. Una **dc** también proporciona un flujo de trabajo de entrega continua básico pero ampliable.

Configuración de compilación (bc)

Define un proceso que se ejecutará en el proyecto OpenShift. Usadas por la característica de fuente a imagen (*source-to-image [S2I]*) de OpenShift para compilar una imagen de contenedor a partir del código fuente de la aplicación almacenado en un servidor Git. Una **bc** funciona junto con una **dc** para proporcionar flujos de trabajo de entrega continua e integración continua básicos pero ampliables.

Rutas

Representan un nombre de host de DNS reconocido por el enrutador de OpenShift como un punto de entrada para aplicaciones y microservicios.

**nota**

Para obtener una lista de todos los recursos disponibles en un clúster de RHOPC y sus abreviaturas, use los comandos **oc api-resources** o **kubectl api-resources**.

Si bien los controladores de replicación de Kubernetes pueden crearse en forma independiente en OpenShift, generalmente son creados por recursos de nivel superior, como los controladores de implementación.

Conexiones en red

Cada contenedor implementado por un clúster de Kubernetes tiene una dirección IP asignada desde una red interna a la que se puede acceder solo desde el nodo que ejecuta el contenedor. Dada la naturaleza efímera del contenedor, las direcciones IP se asignan y se liberan constantemente.

Kubernetes proporciona una *red definida por software (SDN)* que genera las redes internas del contenedor a partir de varios nodos y permite a los contenedores de cualquier pod, dentro de cualquier host, acceder a los pods de otros hosts. El acceso a la SDN solo funciona desde dentro del mismo clúster de Kubernetes.

Los contenedores dentro de los pods de Kubernetes no deben conectarse directamente con la dirección IP dinámica de cada uno. Los servicios resuelven este problema vinculando direcciones IP más estables de la SDN a los pods. Si los pods se reinician, se replican o se reprograman en diferentes nodos, los servicios se actualizan, proporcionando escalabilidad y tolerancia a fallos.

El acceso externo a los contenedores es más complicado. Los servicios de Kubernetes pueden especificar un atributo **NodePort**, que es un puerto de red redirigido por todos los nodos del clúster a la SDN. A continuación, los contenedores en el nodo pueden redirigir un puerto al puerto del nodo. Desafortunadamente, ninguno de estos enfoques se escala bien.

OpenShift hace que el acceso externo a los contenedores sea escalable y más sencillo, mediante la definición de recursos de *route*. Una ruta define nombres y puertos de DNS externos para un servicio. Un enrutador (controlador de ingreso) reenvía las solicitudes HTTP y TLS a las direcciones de servicio dentro de la SDN de Kubernetes. El único requisito es que los nombres de DNS deseados se asignen a las direcciones IP de los nodos de enrutadores de RHOPC.



Referencias

Sitio web de documentación de Kubernetes

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

Sitio web de documentación de OpenShift

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

Comprendión de los operadores

<https://docs.openshift.com/container-platform/4.5/operators/olm-what-operators-are.html>

► Cuestionario

Descripción de Kubernetes y OpenShift

Elija las respuestas correctas para las siguientes preguntas:

- 1. **¿Cuáles son las dos afirmaciones correctas acerca de la arquitectura de Kubernetes? (Elija dos opciones).**
- a. Los nodos de Kubernetes pueden administrarse sin un maestro.
 - b. Los maestros de Kubernetes administran el escalamiento de los pods.
 - c. Los maestros de Kubernetes programan pods para nodos específicos.
 - d. Las herramientas de Kubernetes no pueden usarse para administrar recursos en un clúster de OpenShift.
 - e. Los contenedores creados desde los pods de Kubernetes no pueden administrarse con herramientas independientes como Podman.
- 2. **¿Cuáles son las dos afirmaciones correctas acerca de los tipos de recursos de Kubernetes y OpenShift? (Elija dos opciones).**
- a. Un pod es responsable de aprovisionar su propio almacenamiento persistente.
 - b. Todos los pods generados a partir del mismo controlador de replicación deben ejecutarse en el mismo nodo.
 - c. Una ruta es responsable de proporcionar direcciones IP para el acceso externo a los pods.
 - d. Un controlador de replicación es responsable de monitorear y mantener la cantidad de pods desde una aplicación específica.
- 3. **¿Cuáles son los dos enunciados verdaderos acerca de las redes de Kubernetes y OpenShift? (Elija dos opciones).**
- a. Un servicio de Kubernetes puede proporcionar una dirección IP para acceder a un conjunto de pods.
 - b. Kubernetes es responsable de proporcionar un nombre de dominio completamente calificado para un pod.
 - c. Un controlador de replicación es responsable de enrutar solicitudes externas a los pods.
 - d. Una ruta es responsable de proporcionar nombres de DNS para el acceso externo.

► **4. ¿Cuál es el enunciado correcto acerca del almacenamiento persistente en OpenShift y Kubernetes?**

- a. Un PVC representa un área de almacenamiento que un pod puede usar para almacenar datos y es aprovisionada por el desarrollador de la aplicación.
- b. Un PVC representa un área de almacenamiento que un pod puede solicitar para almacenar datos, pero es aprovisionada por el administrador del clúster.
- c. Un PVC representa la cantidad de memoria que puede asignarse a un nodo, de modo que un desarrollador puede establecer cuánta memoria requiere para ejecutar su aplicación.
- d. Un PVC representa la cantidad de unidades de procesamiento de CPU que pueden asignarse a un pod de aplicación, sujeto a un límite administrado por el administrador del clúster.

► **5. ¿Cuál es el enunciado correcto acerca de las adiciones de OpenShift en relación con Kubernetes?**

- a. OpenShift agrega funciones para simplificar la configuración de Kubernetes de muchos casos prácticos del mundo real.
- b. Las imágenes de contenedores creadas para OpenShift no pueden usarse con Kubernetes simple.
- c. Red Hat mantiene versiones bifurcadas de Kubernetes internas al producto de RHOPC.
- d. Realizar integración continua e implementación continua con RHOPC requiere herramientas externas.

► Solución

Descripción de Kubernetes y OpenShift

Elija las respuestas correctas para las siguientes preguntas:

- 1. **¿Cuáles son las dos afirmaciones correctas acerca de la arquitectura de Kubernetes?** (Elija dos opciones).
- a. Los nodos de Kubernetes pueden administrarse sin un maestro.
 - b. Los maestros de Kubernetes administran el escalamiento de los pods.
 - c. Los maestros de Kubernetes programan pods para nodos específicos.
 - d. Las herramientas de Kubernetes no pueden usarse para administrar recursos en un clúster de OpenShift.
 - e. Los contenedores creados desde los pods de Kubernetes no pueden administrarse con herramientas independientes como Podman.
- 2. **¿Cuáles son las dos afirmaciones correctas acerca de los tipos de recursos de Kubernetes y OpenShift? (Elija dos opciones).**
- a. Un pod es responsable de aprovisionar su propio almacenamiento persistente.
 - b. Todos los pods generados a partir del mismo controlador de replicación deben ejecutarse en el mismo nodo.
 - c. Una ruta es responsable de proporcionar direcciones IP para el acceso externo a los pods.
 - d. Un controlador de replicación es responsable de monitorear y mantener la cantidad de pods desde una aplicación específica.
- 3. **¿Cuáles son los dos enunciados verdaderos acerca de las redes de Kubernetes y OpenShift? (Elija dos opciones).**
- a. Un servicio de Kubernetes puede proporcionar una dirección IP para acceder a un conjunto de pods.
 - b. Kubernetes es responsable de proporcionar un nombre de dominio completamente calificado para un pod.
 - c. Un controlador de replicación es responsable de enrutar solicitudes externas a los pods.
 - d. Una ruta es responsable de proporcionar nombres de DNS para el acceso externo.

► **4. ¿Cuál es el enunciado correcto acerca del almacenamiento persistente en OpenShift y Kubernetes?**

- a. Un PVC representa un área de almacenamiento que un pod puede usar para almacenar datos y es aprovisionada por el desarrollador de la aplicación.
- b. Un PVC representa un área de almacenamiento que un pod puede solicitar para almacenar datos, pero es aprovisionada por el administrador del clúster.
- c. Un PVC representa la cantidad de memoria que puede asignarse a un nodo, de modo que un desarrollador puede establecer cuánta memoria requiere para ejecutar su aplicación.
- d. Un PVC representa la cantidad de unidades de procesamiento de CPU que pueden asignarse a un pod de aplicación, sujeto a un límite administrado por el administrador del clúster.

► **5. ¿Cuál es el enunciado correcto acerca de las adiciones de OpenShift en relación con Kubernetes?**

- a. OpenShift agrega funciones para simplificar la configuración de Kubernetes de muchos casos prácticos del mundo real.
- b. Las imágenes de contenedores creadas para OpenShift no pueden usarse con Kubernetes simple.
- c. Red Hat mantiene versiones bifurcadas de Kubernetes internas al producto de RHOPC.
- d. Realizar integración continua e implementación continua con RHOPC requiere herramientas externas.

Creación de recursos de Kubernetes

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de crear recursos de Kubernetes estándares.

Herramienta de línea de comandos Red Hat OpenShift Container Platform (RHOCP)

El método principal para interactuar con un clúster RHOCP es usar el comando **oc**. El uso básico del comando es a través de sus subcomandos en la siguiente sintaxis:

```
$> oc <command>
```

Antes de interactuar con un clúster, la mayoría de las operaciones requieren un usuario conectado. La sintaxis para iniciar sesión se muestra a continuación:

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

Descripción de la sintaxis de definición de recursos de pod

OpenShift ejecuta contenedores dentro de los pods de Kubernetes; para crear un pod desde una imagen de contenedor, OpenShift necesita una *definición de recursos del pod*. Esta puede ser provista por un archivo de texto JSON o YAML, o puede ser generada a partir de valores predeterminados por el comando **oc new-app** o la consola web de OpenShift.

Un pod es una colección de contenedores y otros recursos. A continuación, se incluye un ejemplo de una definición de pods en un servidor de aplicaciones WildFly en formato YAML:

```
apiVersion: v1
kind: Pod1
metadata:
  name: wildfly2
  labels:
    name: wildfly3
spec:
  containers:
    - resources:
        limits :
          cpu: 0.5
      image: do276/todojee
      name: wildfly
      ports:
        - containerPort: 80804
          name: wildfly
    env:5
```

```
- name: MYSQL_ENV_MYSQL_DATABASE
  value: items
- name: MYSQL_ENV_MYSQL_USER
  value: user1
- name: MYSQL_ENV_MYSQL_PASSWORD
  value: mypa55
```

- ❶ Declara un tipo de recurso de pod de Kubernetes.
- ❷ Un nombre único para un pod en Kubernetes que permite a los administradores ejecutar comandos en él.
- ❸ Crea una etiqueta con una clave denominada **name** que otros recursos en Kubernetes, en general, un servicio, pueden usar para encontrarla.
- ❹ Un atributo que depende del contenedor que identifica el puerto desde el cual se expone el contenedor.
- ❺ Define una colección de variables de entorno.

Algunos pods pueden requerir variables de entorno que puedan ser leídas por un contenedor. Kubernetes transforma todos los pares de **name** y **value** en variables de entorno. Por ejemplo, la variable **MYSQL_ENV_MYSQL_USER** es declarada internamente por el tiempo de ejecución de Kubernetes con un valor denominado **user1**, y se reenvía a la definición de la imagen de contenedor. Dado que el contenedor usa el mismo nombre de variable para obtener el inicio de sesión del usuario, el valor es usado por la instancia del contenedor WildFly para establecer el nombre de usuario que accede a una instancia de base de datos MySQL.

Descripción de la sintaxis de definición de recursos del servicio

Kubernetes proporciona una red virtual que permite que se conecten pods de diferentes trabajadores. Sin embargo, Kubernetes no proporciona ninguna manera fácil para que un pod obtenga las direcciones IP de otros pods.

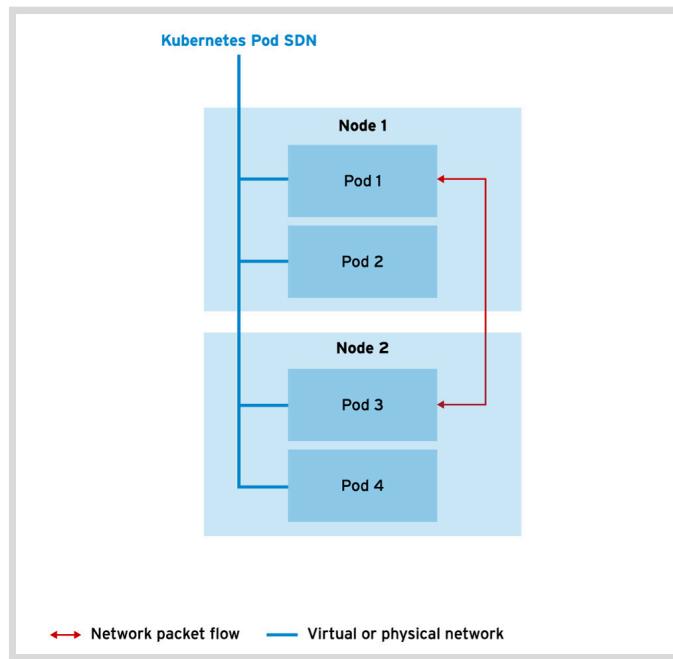


Figura 6.3: Redes básicas de Kubernetes

Los Servicios son recursos esenciales para cualquier aplicación de OpenShift. Permiten que los contenedores en un pod abran conexiones de red con contenedores en otro pod. Un pod puede reiniciarse por muchos motivos y obtiene una dirección IP interna diferente cada vez. En lugar de que un pod tenga que detectar la dirección IP de otro pod después de cada reinicio, un servicio proporciona una dirección IP estable para que usen otros pods, independientemente del nodo de trabajador que ejecute el pod después de cada reinicio.

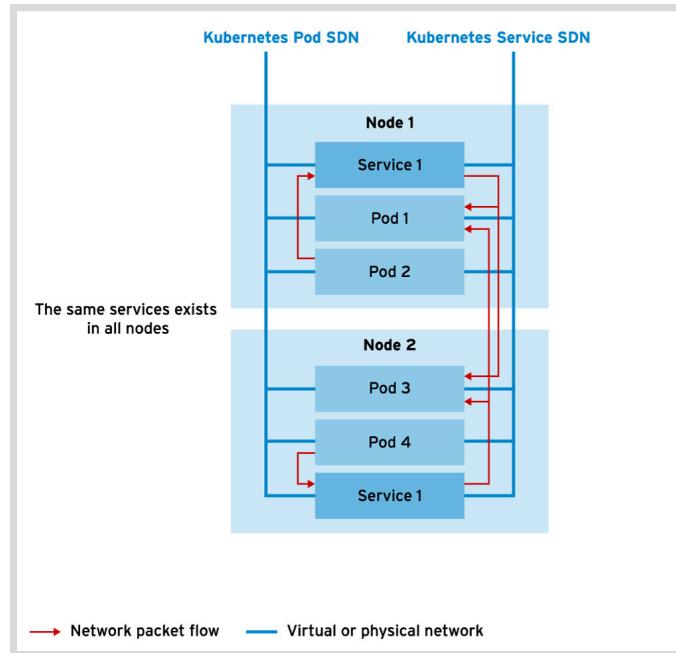


Figura 6.4: Redes de servicios de Kubernetes

La mayoría de las aplicaciones del mundo real no se ejecutan como un solo pod. Deben escalarse horizontalmente, de modo que muchos pods ejecuten los mismos contenedores desde la misma definición de recursos del pod para satisfacer la creciente demanda del usuario. Un servicio está vinculado a un conjunto de pods, y proporciona una dirección IP para todo el conjunto, así como una solicitud de cliente de balanceo de carga entre pods miembros.

El conjunto de los pods que se ejecutan detrás de un servicio es administrado por el recurso *DeploymentConfig*. Un recurso *DeploymentConfig ReplicationController*, que administra la cantidad de copias (réplicas) de pods que deben crearse y crea nuevas si alguna de ellas falla. Los recursos *DeploymentConfig* y *ReplicationController* se explican más adelante en este capítulo.

En la siguiente lista, se muestra una definición de servicio mínimo en sintaxis JSON:

```
{
  "kind": "Service", ①
  "apiVersion": "v1",
  "metadata": {
    "name": "quotedb" ②
  },
  "spec": {
    "ports": [ ③
      {
        "port": 3306,
        "targetPort": 3306
      }
    ],
  }
}
```

```
"selector": {  
    "name": "mysqlDb" ④  
}  
}  
}
```

- ① El tipo de recurso de Kubernetes. En este caso, un Servicio.
- ② Un nombre único para el servicio.
- ③ **ports** (puertos) es un conjunto de objetos que describe puertos de red expuestos por el servicio. El atributo **targetPort** debe coincidir con **containerPort** desde la definición de un contenedor de pods, mientras que el atributo **port** es el puerto que expone el servicio. Los clientes pueden conectarse con el puerto del servicio y el servicio reenvía paquetes al atributo **targetPort** del pod.
- ④ **selector** es el modo en que el servicio busca pods para reenviar paquetes. Los pods de destino deben tener etiquetas que coincidan en sus atributos metadatos. Si el servicio encuentra varios pods con etiquetas que coincidan, balancea la carga de las conexiones de red entre ellos.

Se asigna una dirección IP única a cada servicio para que los clientes se conecten. Esta dirección IP proviene de otro SDN de OpenShift interno, diferente de la red interna de pods, pero visible solo para los pods. Cada pod que coincide con el **selector** se agrega al recurso del servicio como extremo.

Detección de servicios

En general, una aplicación busca una dirección IP y el puerto del servicio mediante el uso de variables de entorno. Para cada servicio dentro de un proyecto de OpenShift, se definen automáticamente las siguientes variables de entorno y se insertan en contenedores para todos los pods dentro del mismo proyecto:

- **SVC_NAME_SERVICE_HOST** es la dirección IP del servicio.
- **SVC_NAME_SERVICE_PORT** es el puerto TCP del servicio.



nota

La parte **SVC_NAME** de la variable se modifica para cumplir con las restricciones de nombre de DNS: las letras van en mayúscula y los guiones bajos (_) se reemplazan por guiones (-).

Otra manera de detectar un servicio desde un pod es mediante el uso de un servidor de DNS interno de OpenShift, que es visible solo para los pods. Se asigna de manera dinámica un registro de SRV a cada servicio con un FQDN que tiene el siguiente formato:

SVC_NAME.PROJECT_NAME.svc.cluster.local

Al detectar servicios con variables de entorno, se debe crear e iniciar un pod solo después de la creación del servicio. Sin embargo, si la aplicación se escribió para detectar servicios con consultas DNS, puede buscar servicios creados después del inicio del pod.

Existen dos maneras para que una aplicación acceda al servicio desde fuera del clúster de OpenShift:

1. Tipo de **NodePort** (Puerto de nodo): este es un enfoque basado en Kubernetes anterior, según el cual el servicio está expuesto a clientes externos mediante la referencia a puertos

disponibles en el host del nodo de trabajador, que, luego, realiza conexiones a través de proxy a la dirección IP del servicio. Usar el comando **oc edit svc** para editar los atributos del servicio y especificar **NodePort** como el valor para **type**, y proporcionar un valor de puerto para el atributo **nodePort**. A continuación, OpenShift proporciona conexiones al servicio a través de la dirección IP pública del host del nodo de trabajo y el valor del puerto establecido en **nodePort**.

2. Rutas de OpenShift: Este es el enfoque preferido en OpenShift para exponer servicios con una URL única. Use el comando **oc expose** para exponer un servicio para acceso externo o exponga un servicio desde la consola web de OpenShift.

En *Figura 6.5*, se muestra cómo los servicios **NodePort** permiten el acceso externo a los servicios de Kubernetes. Las rutas de OpenShift se analizan en más detalle posteriormente en este curso.

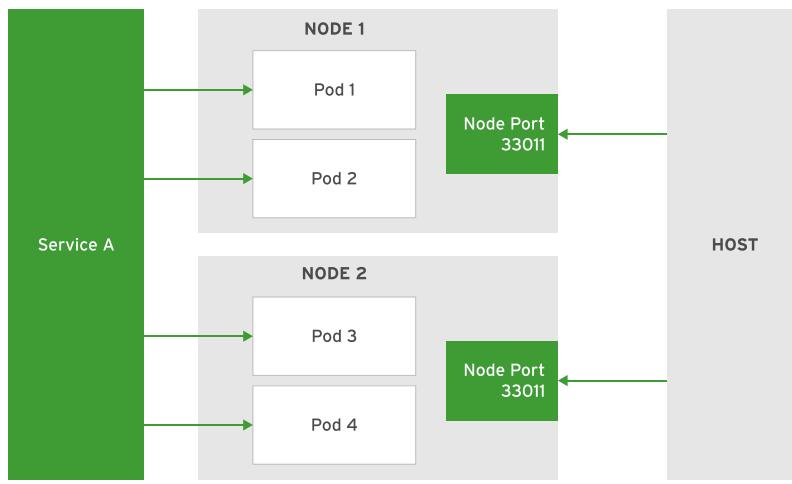


Figura 6.5: Método alternativo para el acceso externo a un servicio de Kubernetes.

OpenShift proporciona el comando **oc port-forward** que reenvía un puerto local a un puerto de pod. Es diferente a tener acceso a un pod a través de un recurso de servicio:

- La asignación de reenvío de puertos existe solo en la estación de trabajo donde se ejecuta el cliente **oc**, mientras que un servicio asigna un puerto a todos los usuarios de red.
- Un servicio balancea la carga de conexiones para potencialmente varios pods, mientras que una asignación de reenvío de puertos reenvía conexiones a un solo pod.



Red Hat desalienta el uso del enfoque **NodePort** para evitar exponer el servicio a conexiones directas. Una asignación vía reenvío de puertos en OpenShift se considera una alternativa más segura.

En el siguiente ejemplo, se demuestra el orden correcto en el cual usar el comando **oc port-forward**:

```
[student@workstation ~]$ oc port-forward mysql-openshift-1-glqrp 3306:3306
```

El comando anterior envía el puerto 3306 de la máquina del desarrollador al puerto 3306 en el pod **db**, donde un servidor MySQL (dentro de un contenedor) acepta las conexiones de red.

**nota**

Al ejecutar este comando, asegúrese de dejar la ventana de terminal en ejecución. Si cierra la ventana o cancela el proceso, se detendrá la asignación de puertos.

Creación de nuevas aplicaciones

Las aplicaciones simples, las aplicaciones complejas de varios niveles y las aplicaciones de microservicio se pueden describir mediante un solo archivo de definición de recursos. Este único archivo contendría muchas definiciones de pods, definiciones de servicios para conectar los pods, controladores de replicación o DeploymentConfigs para escalar horizontalmente los pods de la aplicación, PersistentVolumeClaims para persistir datos de la aplicación y todo lo que se necesite que pueda administrarse mediante OpenShift.

**Importante**

En OpenShift 4.5, de manera predeterminada, el comando **oc new-app** ahora produce recursos Deployment en lugar de recursos DeploymentConfig. Esta versión de DO180 no abarca el recurso Deployment, solo DeploymentConfigs. Para crear recursos DeploymentConfig, puede pasar el indicador **--as-deployment-config** al invocar **oc new-app**, que se realiza en esta versión del curso. Para obtener más información, consulte Comprensión de Deployments y DeploymentConfigs [<https://docs.openshift.com/container-platform/4.5/applications/deployments/what-deployments-are.html#what-deployments-are>].

Se puede usar el comando **oc new-app**, con la opción **-o json** o **-o yaml**, para crear un archivo de definición de recursos esqueleto en formato JSON o YAML, respectivamente. Este archivo se puede personalizar y usar para crear una aplicación con el comando **oc create -f <filename>**, o bien se puede fusionar con otros archivos de definición de recursos para crear una aplicación compuesta.

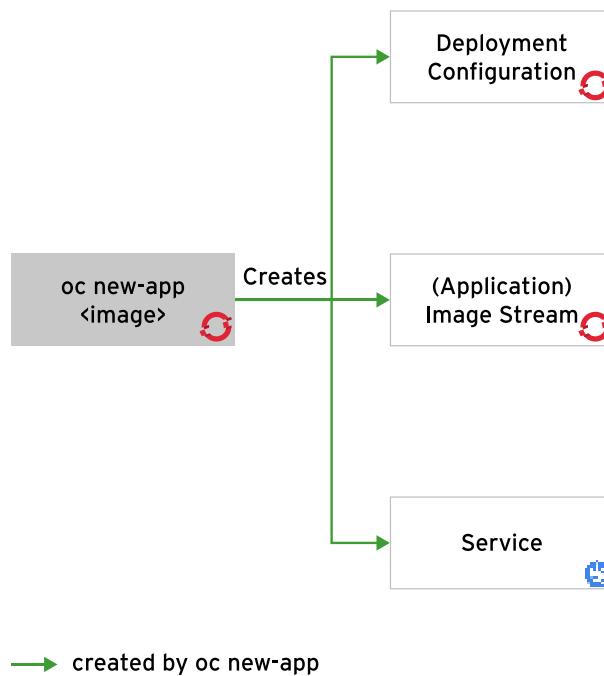
El comando **oc new-app** puede crear pods de aplicaciones para ejecutar en OpenShift de muchas maneras diferentes. Puede crear pods a partir de imágenes de Docker existentes, de Dockerfiles y de un código fuente sin formato mediante el proceso de fuente a imagen (Source-to-Image, S2I).

Ejecute el comando **oc new-app -h** para entender todas las diferentes opciones disponibles para crear nuevas aplicaciones en OpenShift.

El siguiente comando crea una aplicación basada en una imagen, **mysql**, desde Docker Hub, con la etiqueta establecida como **db=mysql**:

```
[student@workstation ~]$ oc new-app mysql --as-deployment-config \
> MYSQL_USER=user MYSQL_PASSWORD=pass MYSQL_DATABASE=testdb -l db=mysql
```

En la siguiente figura, se muestran los recursos de Kubernetes y OpenShift creados por el comando **oc new-app** cuando el argumento es una imagen de contenedor:

**Figura 6.6: Recursos creados para una nueva aplicación**

El siguiente comando crea una aplicación basada en una imagen desde un registro privado de imágenes de Docker:

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

El siguiente comando crea una aplicación basada en un código fuente almacenado en un repositorio Git:

```
oc new-app https://github.com/openshift/ruby-hello-world --name=ruby-hello --as-deployment-config
```

Obtendrá más información acerca del proceso de fuente a imagen (S2I), sus conceptos asociados y maneras más avanzadas de usar **oc new-app** para compilar aplicaciones para OpenShift en la siguiente sección.

Administración de recursos de OpenShift en la línea de comandos

Hay varios comandos esenciales que se usan para administrar recursos de OpenShift, como se describe a continuación.

Use el comando **oc get** para recuperar información acerca de los recursos en el clúster. En general, este comando muestra solo las características más importantes de los recursos y omite información más detallada.

El comando **oc get RESOURCE_TYPE** muestra un resumen de todos los recursos del tipo especificado. A continuación, se ilustra un ejemplo de salida del comando **oc get pods**.

NAME	READY	STATUS	RESTARTS	AGE
nginx-1-5r583	1/1	Running	0	1h
myapp-1-l44m7	1/1	Running	0	1h

oc get all

Use el comando **oc get all** para recuperar un resumen de los componentes más importantes de un clúster. Este comando itera a través de los principales tipos de recursos para el proyecto actual e imprime un resumen de su información.

NAME	DOCKER REPO	TAGS	UPDATED
is/nginx	172.30.1.1:5000/basic-kubernetes/nginx	latest	About an hour ago
dc/nginx			
dc/nginx	1	1	1
Triggered by config,image(nginx:latest)			
rc/nginx-1			
rc/nginx-1	1	1	1
svc/nginx			
svc/nginx	172.30.72.75	<none>	80/TCP, 443/TCP
po/nginx-1-ypp8t			
po/nginx-1-ypp8t	1/1	Running	0
1h			

oc describe RESOURCE RESOURCE_NAME

Si los resúmenes provistos por **oc get** no son suficientes, use el comando **oc describe** para recuperar información adicional. A diferencia del comando **oc get**, no existe ninguna manera para iterar a través de todos los recursos diferentes por tipo. Si bien se puede describir la mayoría de los recursos principales, esta funcionalidad no está disponible en todos los recursos. A continuación, se incluye un ejemplo de salida de la descripción del recurso de un pod:

```
Name: mysql-openshift-1-glqrp
Namespace: mysql-openshift
Priority: 0
PriorityClassName: none
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
        deploymentconfig=mysql-openshift
Annotations: openshift.io/deployment-config.latest-version: 1
            openshift.io/deployment-config.name: mysql-openshift
            openshift.io/deployment.name: mysql-openshift-1
            openshift.io/generated-by: OpenShiftNewApp
            openshift.io/scc: restricted
Status: Running
IP: 10.129.0.85
```

oc get RESOURCE_TYPE RESOURCE_NAME -o yaml

Este comando se puede usar para exportar la definición de un recurso. Los casos de uso típicos incluyen la creación de una copia de seguridad o ayudar a modificar una definición. La opción **-o yaml** imprime la representación del objeto en formato YAML, pero esto se puede modificar al formato JSON con la opción **-o json**.

oc create

Este comando crea recursos a partir de una definición de recursos. En general, se usa junto con el comando **oc get RESOURCE_TYPE RESOURCE_NAME -o yaml** para editar las definiciones.

oc edit

Este comando permite al usuario editar recursos de una definición de recursos. De forma predeterminada, este comando abre un búfer **vi** para editar la definición de recursos.

oc delete RESOURCE_TYPE name

El comando **oc delete** elimina un recurso desde un clúster de OpenShift. Observe que aquí se necesita tener un entendimiento fundamental de la arquitectura de OpenShift, ya que eliminar recursos administrados, como los pods, genera nuevas instancias de aquellos recursos que se crean automáticamente. Cuando se elimina un proyecto, elimina todos los recursos y las aplicaciones contenidos en él.

Comando de opciones de oc exec CONTAINER_ID

El comando **oc exec** ejecuta comandos dentro de un contenedor. Puede usar este comando para ejecutar comandos por lotes interactivos y no interactivos como parte de un script.

Recursos de etiquetado

Cuando se trabaja con muchos recursos en el mismo proyecto, a menudo es útil agrupar esos recursos por aplicación, entorno o algún otro criterio. Para establecer estos grupos, defina etiquetas para los recursos en su proyecto. Las etiquetas son parte de la sección de **metadatos** de un recurso, y se definen como pares de clave-valor, como se muestra en el siguiente ejemplo:

```
apiVersion: v1
kind: Service
metadata:
...contents omitted...
labels:
  app: nexus
  template: nexus-persistent-template
name: nexus
...contents omitted...
```

Muchos subcomandos **oc** soportan una opción **-l** para procesar recursos desde una especificación de etiqueta. Para el comando **oc get**, la opción **-l** actúa como un selector para recuperar solo los objetos que tienen una etiqueta coincidente:

```
$ oc get svc,dc -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      ...
deploymentconfig.apps.openshift.io/nexus  1            1            1            ...
```

**nota**

Si bien cualquier etiqueta puede aparecer en recursos, las claves **app** y **template** son comunes para las etiquetas. Por convención, la clave **app** indica la aplicación relacionada con este recurso. La clave **template** etiqueta todos los recursos generados por la misma plantilla con el nombre de la plantilla.

Cuando se usan plantillas para generar recursos, las etiquetas son especialmente útiles. Un recurso de plantilla tiene una sección **labels** separada de la sección **metadata.labels**. Las etiquetas definidas en la sección **labels** no se aplican a la plantilla en sí, sino que se agregan a cada recurso generado por la plantilla.

```
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...
```

En el ejemplo anterior, se define un recurso de plantilla con una sola etiqueta: **maintainer: redhat**. La plantilla genera un recurso de servicio con tres etiquetas: **app: nexus**, **template: nexus-persistent-template** y **version: 1**.



Referencias

Para obtener información adicional acerca de los servicios y los pods, consulte la sección *Pods y servicios* de la documentación de OpenShift Container Platform:

Arquitectura

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/architecture/index

Para obtener información adicional acerca de la creación de imágenes, consulte la documentación de OpenShift Container Platform:

Creación de imágenes

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/images/index

Las etiquetas y los selectores de etiquetas están disponibles en la sección *Trabajo con objetos Kubernetes* correspondiente a la documentación de Kubernetes:

Etiquetas y selectores

<https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/>

► Ejercicio Guiado

Implementación de un servidor de bases de datos en OpenShift

En este ejercicio, creará e implementará un pod de base de datos MySQL en OpenShift con el comando **oc new-app**.

Resultados

Debería poder crear e implementar un pod de base de datos MySQL en OpenShift.

Andes De Comenzar

En **workstation**, ejecute el siguiente comando para configurar el entorno:

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

► 1. Prepare el entorno del trabajo de laboratorio.

- 1.1. Cargue la configuración de su entorno de aula.

Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 1.2. Inicie sesión en el clúster 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. Cree un nuevo proyecto que contenga su nombre de usuario de desarrollador de RHOCP para los recursos que cree durante este ejercicio:

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

► 2. Cree una nueva aplicación desde la imagen de contenedor **rhscl/mysql-57-rhel7** con el comando **oc new-app**.

Esta imagen requiere que use la opción **-e** para configurar las variables del entorno **MYSQL_USER**, **MYSQL_PASSWORD**, **MYSQL_DATABASE** y **MYSQL_ROOT_PASSWORD**.

Use la opción **--docker-image** con el comando **oc new-app** para especificar la URI del registro privado del aula, de modo que OpenShift no intente extraer la imagen desde Internet:

```
[student@workstation ~]$ oc new-app --as-deployment-config \
> --docker-image=registry.access.redhat.com/rhscl/mysql-57-rhel7:latest \
> --name=mysqlOpenshift \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 -e MYSQL_DATABASE=testdb \
> -e MYSQL_ROOT_PASSWORD=r00tpa55
--> Found Docker image b48e700 (5 weeks old) from registry.access.redhat.com for
"registry.access.redhat.com/rhscl/mysql-57-rhel7:latest"
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "mysqlOpenshift" created
deploymentconfig.apps.openshift.io "mysqlOpenshift" created
service "mysqlOpenshift" created
--> Success
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose svc/mysqlOpenshift'
Run 'oc status' to view your app.
```

- 3. Verifique si el pod de MySQL se creó satisfactoriamente y vea los detalles sobre el pod y su servicio.

- 3.1. Ejecute el comando **oc status** para ver el estado de la nueva aplicación y verificar que la implementación de la imagen MySQL se realizó correctamente:

```
[student@workstation ~]$ oc status
In project youruser-mysql-openshift on server https://
api.cluster.lab.example.com:6443

svc/mysql-openshift - 172.30.114.39:3306
dc/mysql-openshift deploys istag/mysql-openshift:latest
    deployment #1 running for 11 seconds - 0/1 pods
...output omitted...
```

- 3.2. Enumere los pods en este proyecto para verificar que el pod MySQL está listo y ejecutándose:

```
[student@workstation ~]$ oc get pods -o=wide
NAME           READY   STATUS    ...   NODE
mysql-openshift-1-glqrp  1/1     Running   ...   ip-10-0-148-115.ec2.internal
```



nota

Observe el trabajador en el que se está ejecutando el pod. Necesita esta información para poder iniciar sesión posteriormente en el servidor de la base de datos MySQL.

- 3.3. Use el comando **oc describe** para ver más detalles acerca del pod:

```
[student@workstation ~]$ oc describe pod mysql-openshift-1-glqrp
Name:           mysql-openshift-1-glqrp
Namespace:      youruser-mysql-openshift
```

```

Priority:          0
PriorityClassName: <none>
Node:              ip-10-0-148-115.ec2.internal/10.0.148.115
Start Time:        Fri, 15 Feb 2019 02:14:34 +0000
Labels:            app=mysqlOpenshift
                  deployment=mysqlOpenshift-1
                  deploymentconfig=mysqlOpenshift
Annotations:       openshift.io/deployment-config.latest-version: 1
                  openshift.io/deployment-config.name: mysqlOpenshift
                  openshift.io/deployment.name: mysqlOpenshift-1
                  openshift.io/generated-by: OpenShiftNewApp
                  openshift.io/scc: restricted
Status:            Running
IP:                10.129.0.85
...output omitted...

```

- 3.4. Enumere los servicios en este proyecto y verifique que se creó un servicio para acceder al pod de MySQL:

```
[student@workstation ~]$ oc get svc
NAME           TYPE      CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
mysql-openshift ClusterIP  172.30.114.39 <none>        3306/TCP   6m
```

- 3.5. Obtenga los detalles del servicio **mysql-openshift** mediante el comando **oc describe** y note que el tipo de Servicio es **ClusterIP** de forma predeterminada:

```
[student@workstation ~]$ oc describe service mysql-openshift
Name:            mysql-openshift
Namespace:       youruser-mysql-openshift
Labels:          app=mysql-openshift
Annotations:    openshift.io/generated-by: OpenShiftNewApp
Selector:        app=mysql-openshift,deploymentconfig=mysql-openshift
Type:            ClusterIP
IP:              172.30.114.39
Port:            3306-tcp  3306/TCP
TargetPort:      3306/TCP
Endpoints:       10.129.0.85:3306
Session Affinity: None
Events:          <none>
```

- 3.6. Vea detalles sobre la configuración de implementación (**dc**) para esta aplicación:

```
[student@workstation ~]$ oc describe dc mysql-openshift
Name:            mysql-openshift
Namespace:       youruser-mysql-openshift
Created:         15 minutes ago
Labels:          app=mysql-openshift
...output omitted...
Deployment #1 (latest):
  Name:  mysql-openshift-1
  Created: 15 minutes ago
  Status:  Complete
  Replicas: 1 current / 1 desired
```

```
Selector: app=mysql-openshift,deployment=mysql-  
openshift-1,deploymentconfig=mysql-openshift  
Labels: app=mysql-openshift, openshift.io/deployment-config.name=mysql-openshift  
Pods Status: 1 Running / 0 Waiting / 0 Succeeded / 0 Failed  
...output omitted...
```

- 3.7. Exponga el servicio con la creación de una ruta con un nombre predeterminado y un nombre de dominio completamente calificado (FQDN):

```
[student@workstation ~]$ oc expose service mysql-openshift  
route.route.openshift.io/mysql-openshift exposed  
[student@workstation ~]$ oc get routes  
NAME           HOST/PORT          ... PORT  
mysql-openshift mysql-openshift-youruser-mysql-openshift... 3306-tcp
```

- 4. Conecte el servidor de la base de datos MySQL y verifique que la base de datos se creó satisfactoriamente.

- 4.1. Desde la máquina **workstation**, configure el reenvío de puertos entre **workstation** y el pod de la base de datos que se ejecuta en OpenShift usando el puerto 3306. El terminal se colgará después de ejecutar el comando.

```
[student@workstation ~]$ oc port-forward mysql-openshift-1-glqrp 3306:3306  
Forwarding from 127.0.0.1:3306 -> 3306  
Forwarding from [::1]:3306 -> 3306
```

- 4.2. Desde la máquina **workstation**, abra otro terminal y conéctelo al servidor MySQL mediante el cliente MySQL.

```
[student@workstation ~]$ mysql -uuser1 -pmypa55 --protocol tcp -h localhost  
Welcome to the MariaDB monitor. Commands end with ; or \g.  
Your MySQL connection id is 1  
Server version: 5.6.34 MySQL Community Server (GPL)  
  
Copyright (c) 2000, 2016, Oracle, MariaDB Corporation Ab and others.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
MySQL [(none)]>
```

- 4.3. Verifique la creación de la base de datos **testdb**.

```
MySQL [(none)]> show databases;  
+-----+  
| Database      |  
+-----+  
| information_schema |  
| testdb         |  
+-----+  
2 rows in set (0.00 sec)
```

- 4.4. Salga del prompt de MySQL:

```
MySQL [(none)]> exit  
Bye
```

Cierre el terminal y regrese al anterior. Presione **Ctrl+C** para finalizar el proceso de reenvío de puertos.

```
[student@workstation ~]$ oc port-forward mysql-openshift-1-5qh98 3306:3306  
Forwarding from 127.0.0.1:3306 -> 3306  
Forwarding from [::1]:3306 -> 3306  
Handling connection for 3306  
^C[student@workstation ~]$
```

- 5. Elimine el proyecto, para eliminar todos los recursos dentro del proyecto:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-mysql-openshift
```

Finalizar

En **workstation**, ejecute el script **lab openshift-resources finish** para terminar este trabajo de laboratorio.

```
[student@workstation ~]$ lab openshift-resources finish
```

Esto concluye el ejercicio.

Creación de rutas

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de verificar los servicios usando rutas de OpenShift.

Trabajo con rutas

Los servicios permiten el acceso de red entre pods dentro de una instancia de OpenShift, y las rutas permiten el acceso de red a pods desde los usuarios y las aplicaciones fuera de la instancia de OpenShift.

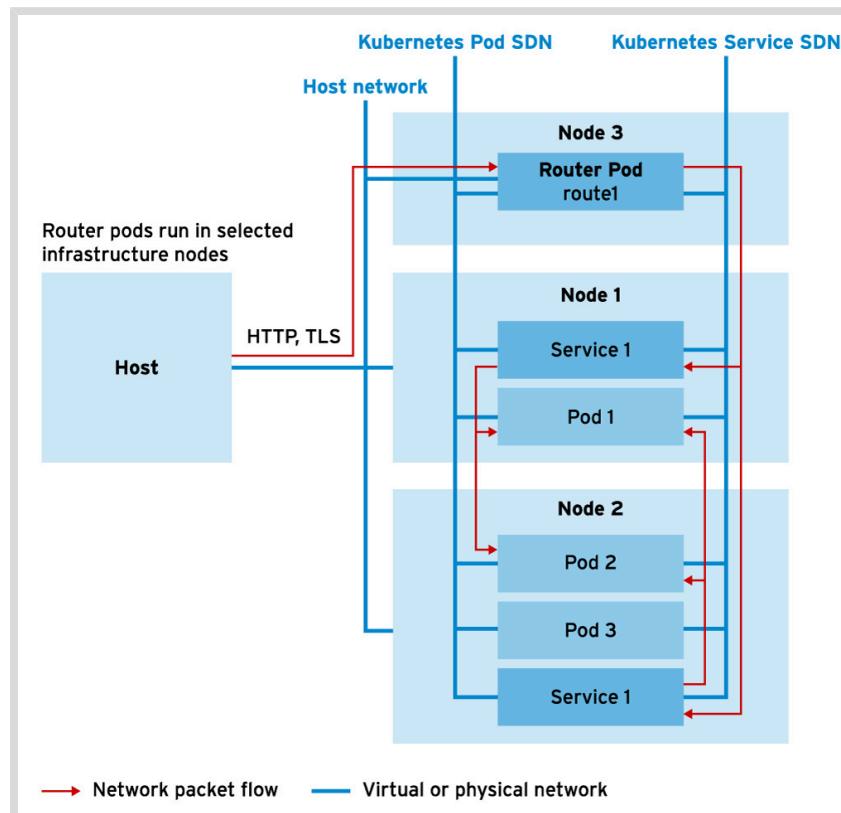


Figura 6.7: Las rutas de OpenShift y los servicios de Kubernetes

Una ruta conecta una dirección IP de acceso público y un nombre de host de DNS a una IP de servicio de acceso interno. Usa el recurso de servicios para encontrar los extremos, es decir, los puertos expuestos por el servicio.

Las rutas de OpenShift son implementadas por un servicio de enruteador que abarca todo el clúster, que se ejecuta como una aplicación en contenedor en el clúster de OpenShift. OpenShift escala y replica los pods del enruteador como cualquier otra aplicación OpenShift.

**nota**

En la práctica, para mejorar el rendimiento y reducir la latencia, el enrutador de OpenShift se conecta directamente a los pods usando la red definida por software (SDN) del pod interno.

El servicio de enrutador usa *HAProxy* como la implementación predeterminada.

Una consideración importante para los administradores de OpenShift es que los nombres de hosts de DNS pública configurados para las rutas deben apuntar a las direcciones IP de acceso público de los nodos que ejecutan el enrutador. Los pods de enrutadores, a diferencia de los pods de aplicaciones regulares, se enlazan con las direcciones IP públicas de los nodos en lugar de con la SDN de pods interna.

En el siguiente ejemplo, se muestra una ruta mínima definida con la sintaxis JSON:

```
{
  "apiVersion": "v1",
  "kind": "Route",
  "metadata": {
    "name": "quoteapp"
  },
  "spec": {
    "host": "quoteapp.apps.example.com",
    "to": {
      "kind": "Service",
      "name": "quoteapp"
    }
  }
}
```

Los atributos **apiVersion**, **kind** y **metadata** siguen las reglas estándares de definición de recursos de Kubernetes. El valor **Route** para **kind** muestra que este es un recurso de ruta, y el atributo **metadata.name** le da a esta ruta específica el identificador **quoteapp**.

Como sucede con los pods y los servicios, la parte principal es el atributo **spec**, que es un objeto que contiene los siguientes atributos:

- **host** es una cadena que contiene el nombre FQDN asociado con la ruta. DNS debe resolver este FQDN a la dirección IP del enrutador OpenShift. Los detalles para modificar la configuración de DNS están fuera del alcance de este curso.
- **to** es un objeto que indica el recurso al que apunta esta ruta. En este caso, la ruta apunta a un servicio OpenShift con **name** configurado en **quoteapp**.

**nota**

Los nombres de diferentes tipos de recursos no están en conflicto. Es perfectamente legal tener una ruta denominada **quoteapp**, que apunta a un servicio también nombrado **quoteapp**.

**Importante**

A diferencia de los servicios, que usan selectores para vincularse a recursos de pods que contienen etiquetas específicas, una ruta se vincula directamente al nombre del recurso del servicio.

Creación de rutas

Use el comando **oc create** para crear recursos de ruta, al igual que cualquier otro recurso de OpenShift. Debe proporcionar un archivo de definición de recursos JSON o YAML, que define la ruta, al comando **oc create**.

El comando **oc new-app** no crea un recurso de ruta cuando compila un pod a partir de imágenes de contenedores, Dockerfiles o código fuente de la aplicación. Después de todo, **oc new-app** no sabe si se pretende que se pueda acceder al pod desde fuera de la instancia de OpenShift.

Otra manera de crear una ruta es usar el comando **oc expose service** y pasar un nombre de recurso del servicio como entrada. La opción **--name** se puede usar para controlar el nombre del recurso de ruta. Por ejemplo:

```
$ oc expose service quotedb --name quote
```

De manera predeterminada, las rutas creadas por **oc expose** generan nombres DNS de la forma:

route-name-project-name.default-domain

Donde:

- *route-name* es el nombre asignado a la ruta. Si no se establece un nombre explícito, OpenShift asigna a la ruta el mismo nombre que el recurso de origen (por ejemplo, el nombre del servicio).
- *project-name* es el nombre del proyecto que contiene el recurso.
- *default-domain* está configurado en el maestro de OpenShift y corresponde al dominio de DNS comodín detallado como requisito previo para instalar OpenShift.

Por ejemplo, cuando se crea una ruta con el nombre **quote** en un proyecto denominado **test** desde una instancia de OpenShift donde el dominio comodín es **clouddapps.example.com**, se genera el FQDN **quote-test.clouddapps.example.com**.

**nota**

El servidor de DNS que aloja al dominio comodín desconoce los nombres de host de rutas. Simplemente resuelve todo nombre a las direcciones IP configuradas.

Solo el enrutador de OpenShift conoce acerca de los nombres de host de la ruta, y trata a cada uno como un host virtual HTTP. Los nombres de dominio comodín no válidos que no se corresponden con ninguna ruta son bloqueados por el enrutador de OpenShift y generan un error HTTP 404.

Aprovechamiento del servicio de enrutamiento predeterminado

El servicio de enrutamiento predeterminado se implementa como un pod **HAProxy**. Los pods y los contenedores del enrutador, y su configuración, se pueden inspeccionar como cualquier otro recurso en un clúster de OpenShift:

```
$ oc get pod --all-namespaces -l app=router
NAMESPACE          NAME           READY   STATUS    RESTARTS   AGE
openshift-ingress  router-default-746b5cfb65-f6sdm 1/1     Running   1          4d
```

De manera predeterminada, el enrutador se implementa en el proyecto **openshift-ingress**. Use el comando **oc describe pod** para obtener los detalles de configuración de enrutamiento:

```
$ 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...
```

El subdominio, o el dominio predeterminado que se usará en todas las rutas predeterminadas, toma su valor de la entrada **ROUTER_CANONICAL_HOSTNAME**.



Referencias

Para obtener información adicional sobre la arquitectura de rutas en OpenShift, consulte las secciones *Arquitectura* y *Guía para desarrolladores* de **Documentación de OpenShift Container Platform**.

https://access.redhat.com/documentation/en-us/openshift_container_platform/

► Ejercicio Guiado

Exposición de un servicio como una ruta

En este ejercicio, creará, compilará e implementará una aplicación en un clúster de OpenShift y expondrá su servicio como una ruta.

Resultados

Debería poder exponer un servicio como una ruta para una aplicación de OpenShift implementada.

Andes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

- 1. Prepare el entorno del trabajo de laboratorio.

- 1.1. Cargue la configuración de su entorno de aula.

Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 1.2. Inicie sesión en el clúster 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. Cree un nuevo proyecto que contenga su nombre de usuario de desarrollador de RHOCUP para los recursos que cree durante este ejercicio:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-route
```

- 2. Cree una nueva aplicación PHP con la fuente a imagen desde el directorio **php-helloworld** en el repositorio de Git en <http://github.com/yourgituser/D0180-apps/>.

- 2.1. Use el comando **oc new-app** para crear la aplicación PHP.

**Importante**

En el siguiente ejemplo, se usa una barra diagonal invertida (\) para indicar que la segunda línea es una continuación de la primera. Si desea ignorar la barra diagonal invertida, puede ingresar todo el comando en una línea.

```
[student@workstation ~]$ oc new-app --as-deployment-config \
> php:7.3~https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps \
> --context-dir php-helloworld --name php-helloworld
--> Found image fbe3911 (13 days old) in image stream "openshift/php" under tag
    "7.3" for "php:7.3"
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
Build scheduled, use 'oc logs -f bc/php-helloworld' 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/php-helloworld'
Run 'oc status' to view your app.
```

- 2.2. Espere hasta que la aplicación finalice la implementación y compilación mediante el monitoreo del progreso con el comando **oc get pods -w**:

```
[student@workstation ~]$ oc get pods -w
NAME             READY   STATUS    RESTARTS   AGE
php-helloworld-1-build  0/1     Init:0/2   0          2s
php-helloworld-1-build  0/1     Init:0/2   0          4s
php-helloworld-1-build  0/1     Init:1/2   0          5s
php-helloworld-1-build  0/1     PodInitializing   0          6s
php-helloworld-1-build  1/1     Running   0          7s
php-helloworld-1-deploy  0/1     Pending    0          0s
php-helloworld-1-deploy  0/1     Pending    0          0s
php-helloworld-1-deploy  0/1     ContainerCreating   0          0s
php-helloworld-1-build  0/1 Completed  0          5m8s
php-helloworld-1-cnphm  0/1     Pending    0          0s
php-helloworld-1-cnphm  0/1     Pending    0          1s
php-helloworld-1-deploy  1/1     Running   0          4s
php-helloworld-1-cnphm  0/1     ContainerCreating   0          1s
php-helloworld-1-cnphm  1/1 Running  0          62s
php-helloworld-1-deploy  0/1     Completed   0          65s
php-helloworld-1-deploy  0/1     Terminating 0          66s
php-helloworld-1-deploy  0/1     Terminating 0          66s
^C
```

El resultado exacto puede diferir en nombres, estado, tiempo y orden. Busque el contenedor **Completed** (Completado) con el sufijo **deploy**: eso significa que la aplicación se implementó correctamente. El contenedor en estado **Running** (En ejecución) con un sufijo aleatorio (**cnphm** en el ejemplo) contiene la aplicación y la muestra activa y en ejecución.

Como alternativa, monitoree los registros de compilación e implementación con los comandos **oc logs -f bc/php-helloworld** y **oc logs -f dc/php-helloworld**, respectivamente.

```
[student@workstation ~]$ oc logs -f bc/php-helloworld
Cloning "https://github.com/yourgituser/D0180-apps" ...
Commit: f7cd8963ef353d9173c3a21dccf402f3616840b (Initial commit, including all
apps previously in course)
...output omitted...
STEP 7: USER 1001
STEP 8: RUN /usr/libexec/s2i/assemble
--> Installing application source...
...output omitted...
Push successful
[student@workstation ~]$ oc logs -f dc/php-helloworld
=> sourcing 50-mpm-tuning.conf ...
=> sourcing 40-ssl-certs.sh ...
...output omitted...
[core:notice] [pid 1] AH00094: Command line: 'httpd -D FOREGROUND'
```

Su resultado exacto puede diferir.

2.3. Revise el servicio para esta aplicación con el comando **oc describe**:

```
[student@workstation ~]$ oc describe svc/php-helloworld
Name:           php-helloworld
Namespace:      youruser-route
Labels:         app=php-helloworld
Annotations:   openshift.io/generated-by=OpenShiftNewApp
Selector:       app=php-helloworld,deploymentconfig=php-helloworld
Type:          ClusterIP
IP:            172.30.200.65
Port:          8080-tcp  8080/TCP
TargetPort:    8080/TCP
Endpoints:    10.129.0.31:8080
Port:          8443-tcp  8443/TCP
TargetPort:    8443/TCP
Endpoints:    10.129.0.31:8443
Session Affinity: None
Events:        <none>
```

La dirección IP que se muestra en la salida del comando puede diferir.

- ▶ 3. Exponga el servicio, que cree una ruta: Use el nombre predeterminado y el nombre de dominio completo (FQDN) para la ruta:

```
[student@workstation ~]$ oc expose svc/php-helloworld
route.route.openshift.io/php-helloworld exposed
[student@workstation ~]$ oc describe route
Name:           php-helloworld
Namespace:      youruser-route
Created:       4 minutes ago
Labels:         app=php-helloworld
Annotations:   openshift.io/host.generated=true
```

```
Requested Host: php-helloworld-youruser-routes.your_wildcard_domain
    exposed on router default (host your_wildcard_domain) 4 minutes ago
Path: <none>
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port: 8080-tcp

Service: php-helloworld
Weight: 100 (100%)
Endpoints: 10.130.0.48:8443, 10.130.0.48:8080
```

- 4. Acceda al servicio desde un host externo al clúster para verificar que el servicio y la ruta funcionan.

```
[student@workstation ~]$ curl php-helloworld-${RHT_OCP4_DEV_USER}-route.
${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.11
```



nota

La salida de la aplicación PHP depende del código real en el repositorio Git. Puede ser diferente si actualiza el código en las secciones anteriores.

Note que el FQDN se compone del nombre de la aplicación y el nombre del proyecto de forma predeterminada. El resto del FQDN, el subdominio, se define al instalarse OpenShift.

- 5. Reemplace esta ruta con una ruta denominada **xyz**.

- 5.1. Elimine la ruta actual:

```
[student@workstation ~]$ oc delete route/php-helloworld
route.route.openshift.io "php-helloworld" deleted
```



nota

Eliminar la ruta es opcional. Puede tener varias rutas para el mismo servicio, siempre que tengan nombres diferentes.

- 5.2. Cree una ruta para el servicio con el nombre **youruser-xyz**.

```
[student@workstation ~]$ oc expose svc/php-helloworld \
> --name=${RHT_OCP4_DEV_USER}-xyz
route.route.openshift.io/youruser-xyz exposed
[student@workstation ~]$ oc describe route
Name: youruser-xyz
Namespace: youruser-route
Created: About a minute ago
Labels: app=php-helloworld
Annotations: openshift.io/host.generated=true
Requested Host: youruser-xyz-youruser-route.your_wildcard_domain
    exposed on router default (host your_wildcard_domain) 2 minutes ago
```

```
Path: <none>
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port: 8080-tcp

Service: php-helloworld
Weight: 100 (100%)
Endpoints: 10.130.0.48:8443, 10.130.0.48:8080
```

Tenga en cuenta el nuevo FQDN que se generó en función del nuevo nombre de la ruta. Tanto el nombre de la ruta como el nombre del proyecto contienen su nombre de usuario; por ende, aparece dos veces en el FQDN de la ruta.

- 5.3. Realice una solicitud HTTP con el FQDN en el puerto 80:

```
[student@workstation ~]$ curl \
> ${RHT_OCP4_DEV_USER}-xyz-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.11
```

Finalizar

En **workstation**, ejecute el script **lab openshift-routes finish** para terminar este ejercicio.

```
[student@workstation ~]$ lab openshift-routes finish
```

Esto concluye el ejercicio guiado.

Creación de aplicaciones con Fuente a imagen (Source-to-Image)

Objetivos

Después de completar esta sección, los estudiantes deberían poder implementar una aplicación con la herramienta Fuente a imagen (Source-to-Image, S2I) de OpenShift Container Platform.

Proceso de fuente a imagen (S2I)

Fuente a imagen (S2I) es una herramienta que facilita la compilación de una imagen de contenedor desde un código fuente de la aplicación. Esta herramienta toma el código fuente de una aplicación desde un repositorio Git, inserta el código fuente en un contenedor base según el lenguaje y el marco (framework) deseados, y produce una nueva imagen de contenedor que ejecuta la aplicación ensamblada.

En *Figura 6.8*, se muestran los recursos creados por el comando **oc new-app** cuando el argumento es un repositorio de código fuente de aplicación. Observe que S2I también crea una configuración de implementación y todos sus recursos dependientes.

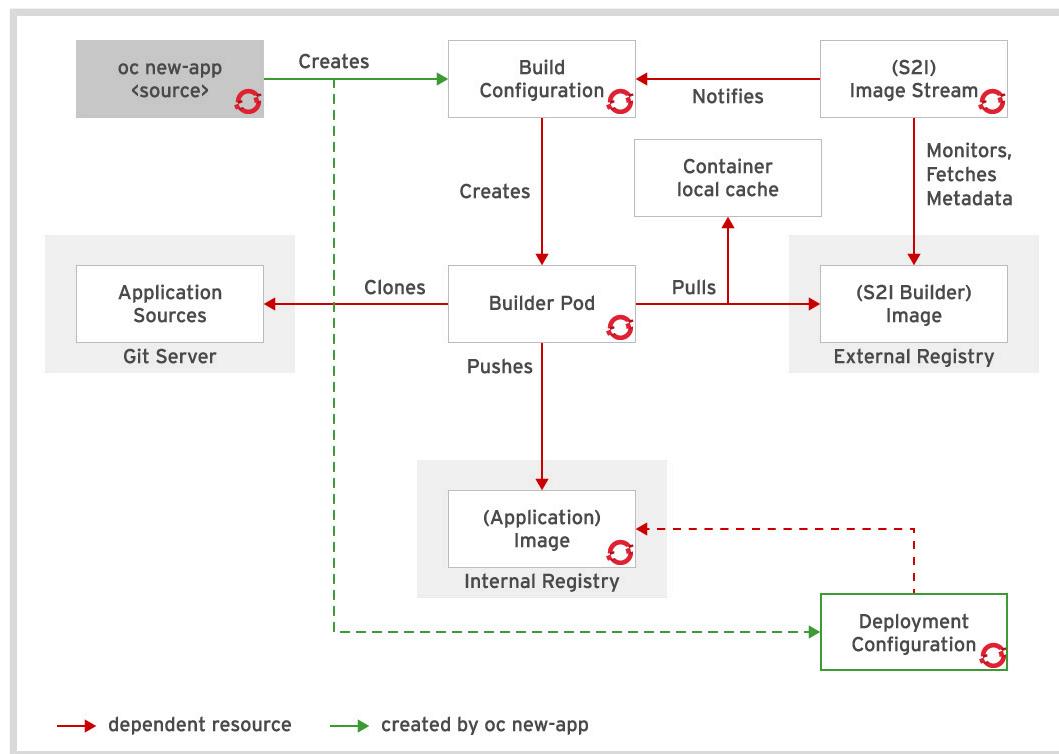


Figura 6.8: Configuración de implementación y recursos dependientes

S2I es la estrategia principal usada para la compilación de aplicaciones en OpenShift Container Platform. A continuación se detallan los principales motivos para usar compilaciones fuente:

- Eficiencia del usuario: no es necesario que los desarrolladores entiendan Dockerfiles ni comandos del sistema operativo como **yum install**. Trabajan con sus herramientas de lenguaje de programación estándar.

- Aplicación de parches: S2I permite recompilar todas las aplicaciones de manera consistente si una imagen base necesita un parche debido a un problema de seguridad. Por ejemplo, si se encuentra un problema de seguridad en una imagen base PHP, la actualización de esta imagen con los parches de seguridad actualiza todas las aplicaciones que usan esta imagen como base.
- Velocidad: Con S2I, el proceso de ensamblaje puede realizar una gran cantidad de operaciones complejas sin crear una nueva capa en cada paso, lo que genera compilaciones más rápidas.
- Ecosistema: S2I impulsa un ecosistema compartido de imágenes donde las imágenes base y los scripts pueden personalizarse y reutilizarse en varios tipos de aplicaciones.

Descripción de flujos de imágenes

OpenShift implementa nuevas versiones de aplicaciones de usuario en pods rápidamente. Para crear una nueva aplicación, además del código fuente de la aplicación, se requiere una imagen base (la imagen del compilador S2I). Si se actualiza cualquiera de estos dos componentes, OpenShift crea una nueva imagen de contenedor. Los pods creados con la imagen de contenedor anterior se reemplazan con pods que usan la nueva imagen.

Si bien es evidente que la imagen de contenedor se debe actualizar cuando se modifica el código de la aplicación, es posible que no sea evidente que los pods implementados también se deben actualizar si se modifica la imagen del compilador.

El *recurso del flujo de imágenes* es una configuración que denomina imágenes de contenedores específicas asociadas con *etiquetas de flujo de imágenes*, un alias para estas imágenes de contenedores. OpenShift compila aplicaciones respecto del flujo de imágenes. El instalador de OpenShift ingresa varios flujos de imágenes de forma predeterminada durante la instalación.

Para determinar los flujos de imágenes disponibles, use el comando **oc get**, como se muestra a continuación:

```
$ oc get is -n openshift
NAME          IMAGE REPOSITORY           TAGS
cli           ...svc:5000/openshift/cli   latest
dotnet        ...svc:5000/openshift/dotnet 2.0,2.1,latest
dotnet-runtime ...svc:5000/openshift/dotnet-runtime 2.0,2.1,latest
httpd         ...svc:5000/openshift/httpd    2.4,latest
jenkins       ...svc:5000/openshift/jenkins 1,2
mariadb       ...svc:5000/openshift/mariadb 10.1,10.2,latest
mongodb       ...svc:5000/openshift/mongodb 2.4,2.6,3.2,3.4,3.6,latest
mysql         ...svc:5000/openshift/mysql   5.5,5.6,5.7,latest
nginx         ...svc:5000/openshift/nginx   1.10,1.12,1.8,latest
nodejs        ...svc:5000/openshift/nodejs  0.10,10,11,4,6,8,latest
perl          ...svc:5000/openshift/perl   5.16,5.20,5.24,5.26,latest
php           ...svc:5000/openshift/php    5.5,5.6,7.0,7.1,latest
postgresql    ...svc:5000/openshift/postgresql 10,9.2,9.4,9.5,9.6,latest
python        ...svc:5000/openshift/python  2.7,3.3,3.4,3.5,3.6,latest
redis         ...svc:5000/openshift/redis   3.2,latest
ruby          ...svc:5000/openshift/ruby   2.0,2.2,2.3,2.4,2.5,latest
wildfly       ...svc:5000/openshift/wildfly 10.0,10.1,11.0,12.0,...
```



nota

Su instancia de OpenShift puede tener más o menos flujos de imágenes dependiendo de las adiciones locales y las versiones complementarias de OpenShift.

OpenShift detecta cuando un flujo de imágenes cambia y puede actuar en función de ese cambio. Si surge un problema de seguridad en la imagen **nodejs-010-rhel7**, puede actualizarse en el repositorio de imágenes y OpenShift puede desencadenar automáticamente una nueva compilación del código de la aplicación.

Una organización probablemente elegirá varias imágenes S2I base soportadas de Red Hat, pero también podrá crear sus propias imágenes base.

Compilación de una aplicación con S2I y la CLI

La compilación de una aplicación con S2I se puede lograr con la CLI de OpenShift.

Una aplicación se puede crear con el proceso S2I con el comando **oc new-app** desde la CLI.

```
$ oc new-app ① --as-deployment-config ② php~http://my.git.server.com/my-app③ --  
name=myapp④
```

- ②** Cree un recurso DeploymentConfig en lugar de un recurso Deployment.
- ②** El flujo de imágenes usado en el proceso aparece en el lado izquierdo de la tilde (~).
- ③** La URL después de la tilde indica la ubicación del repositorio Git del código fuente.
- ④** Configura el nombre de la aplicación.



nota

En lugar de usar la tilde, puede configurar el flujo de imágenes usando la opción **-i**.

```
$ oc new-app --as-deployment-config -i php http://services.lab.example.com/app  
--name=myapp
```

El comando **oc new-app** permite la creación de aplicaciones con el código fuente desde un repositorio Git local o remoto. Si solo se especifica un repositorio fuente, **oc new-app** intenta identificar el flujo de imágenes correcto para usar en la compilación de la aplicación. Además del código de aplicación, S2I también puede identificar y procesar Dockerfiles para crear una nueva imagen.

En el siguiente ejemplo, se crea una aplicación con el repositorio Git en el directorio actual.

```
$ oc new-app --as-deployment-config .
```



Importante

Al usar un repositorio Git local, el repositorio debe tener un origen remoto que indique una URL a la que pueda acceder la instancia de OpenShift.

También se puede crear una aplicación con un repositorio Git remoto y un subdirectorio de contexto:

```
$ oc new-app --as-deployment-config \  
https://github.com/openshift/sti-ruby.git \  
--context-dir=2.0/test/puma-test-app
```

Finalmente, se puede crear una aplicación con un repositorio Git remoto con una referencia de ramificación específica:

```
$ oc new-app --as-deployment-config \
https://github.com/openshift/ruby-hello-world.git#beta4
```

Si no se especifica un flujo de imágenes en el comando, **new-app** intenta determinar qué compilador de lenguajes se debe usar según la presencia de ciertos archivos en el directorio root del repositorio:

Lenguaje	Archivos
Ruby	Rakefile Gemfile config.ru
Java EE	pom.xml
Node.js	app.json package.json
PHP	index.php composer.json
Python	requirements.txt config.py
Perl	index.pl cpanfile

Después de que se detecte un lenguaje, el comando **new-app** busca etiquetas de flujos de imágenes que soporten el lenguaje detectado o un flujo de imágenes que coincida con el nombre del lenguaje detectado.

Cree un archivo de definición de recursos JSON con el parámetro **-o json** y la redirección de salida:

```
$ oc -o json new-app --as-deployment-config \
> php-http://services.lab.example.com/app \
> --name=myapp > s2i.json
```

Este archivo de definición JSON crea una lista de recursos. El primer recurso es el flujo de imágenes:

```
...output omitted...
{
  "kind": "ImageStream", ①
  "apiVersion": "image.openshift.io/v1",
  "metadata": {
    "name": "myapp", ②
    "creationTimestamp": null
    "labels": {
      "app": "myapp"
    },
    "annotations": {
      "openshift.io/generated-by": "OpenShiftNewApp"
    }
  },
  "spec": {
    "lookupPolicy": {
```

```

        "local": false
    },
},
"status": {
    "dockerImageRepository": ""
}
},
...output omitted...

```

- ➊ Defina un tipo de recurso de flujo de imágenes.
- ➋ Asigne el nombre **myapp** al flujo de imágenes.

La configuración de compilación (**bc**) debe definir parámetros de entrada y desencadenadores que se ejecutan para transformar el código fuente en una imagen ejecutable. **BuildConfig** (BC) es el segundo recurso y los siguientes ejemplos brindan una descripción general de los parámetros usados por OpenShift para crear una imagen ejecutable.

```

...output omitted...
{
    "kind": "BuildConfig", ➊
    "apiVersion": "build.openshift.io/v1",
    "metadata": {
        "name": "myapp", ➋
        "creationTimestamp": null,
        "labels": {
            "app": "myapp"
        },
        "annotations": {
            "openshift.io/generated-by": "OpenShiftNewApp"
        }
    },
    "spec": {
        "triggers": [
            {
                "type": "GitHub",
                "github": {
                    "secret": "S5_4BZpPabM6KrIuPBvI"
                }
            },
            {
                "type": "Generic",
                "generic": {
                    "secret": "3q8K8JNDoRzhj0z1KgMz"
                }
            },
            {
                "type": "ConfigChange"
            },
            {
                "type": "ImageChange",
                "imageChange": {}
            }
        ],
        "source": {

```

```

    "type": "Git",
    "git": {
        "uri": "http://services.lab.example.com/app" 3
    }
},
"strategy": {
    "type": "Source", 4
    "sourceStrategy": {
        "from": {
            "kind": "ImageStreamTag",
            "namespace": "openshift",
            "name": "php:7.3" 5
        }
    }
},
"output": {
    "to": {
        "kind": "ImageStreamTag",
        "name": "myapp:latest" 6
    }
},
"resources": {},
"postCommit": {},
"nodeSelector": null
},
"status": {
    "lastVersion": 0
}
},
...output omitted...

```

- 1** Defina un tipo de recurso de **BuildConfig**.
- 2** Asigne el nombre **myapp** a **BuildConfig**.
- 3** Defina la dirección del repositorio Git de código fuente.
- 4** Defina la estrategia para usar S2I.
- 5** Defina la imagen del compilador como el flujo de imágenes php:7.3.
- 6** Asigne el nombre **myapp:latest** al flujo de imágenes de salida.

El tercer recurso es la configuración de implementación que debe personalizar el proceso de implementación en OpenShift. Puede incluir parámetros y desencadenadores necesarios para crear nuevas instancias de contenedores y se traducen en un controlador de reproducción desde Kubernetes. A continuación, se detallan algunas de las características provistas por los objetos **DeploymentConfig**:

- Estrategias personalizables del usuario para la transición de las implementaciones existentes a las nuevas implementaciones.
- Reversiones a una implementación anterior.
- Escalamiento manual de replicaciones.

```

...output omitted...
{
    "kind": "DeploymentConfig", 1
    "apiVersion": "apps.openshift.io/v1",

```

```

"metadata": {
    "name": "myapp", ❷
    "creationTimestamp": null,
    "labels": {
        "app": "myapp"
    },
    "annotations": {
        "openshift.io/generated-by": "OpenShiftNewApp"
    }
},
"spec": {
    "strategy": {
        "resources": {}
    },
    "triggers": [
        {
            "type": "ConfigChange" ❸
        },
        {
            "type": "ImageChange", ❹
            "imageChangeParams": {
                "automatic": true,
                "containerNames": [
                    "myapp"
                ],
                "from": {
                    "kind": "ImageStreamTag",
                    "name": "myapp:latest"
                }
            }
        }
    ],
    "replicas": 1,
    "test": false,
    "selector": {
        "app": "myapp",
        "deploymentconfig": "myapp"
    },
    "template": {
        "metadata": {
            "creationTimestamp": null,
            "labels": {
                "app": "myapp",
                "deploymentconfig": "myapp"
            },
            "annotations": {
                "openshift.io/generated-by": "OpenShiftNewApp"
            }
        },
        "spec": {
            "containers": [
                {
                    "name": "myapp",
                    "image": "myapp:latest", ❺
                    "ports": [ ❻

```

```

        {
          "containerPort": 8080,
          "protocol": "TCP"
        },
        {
          "containerPort": 8443,
          "protocol": "TCP"
        }
      ],
      "resources": {}
    }
  ]
}
},
"status": {
  "latestVersion": 0,
  "observedGeneration": 0,
  "replicas": 0,
  "updatedReplicas": 0,
  "availableReplicas": 0,
  "unavailableReplicas": 0
}
},
...output omitted...

```

- ① Defina un tipo de recurso de **DeploymentConfig**.
- ② Asigne el nombre **myapp** a **DeploymentConfig**.
- ③ Un desencadenador de cambios de configuración genera la creación de una nueva implementación cada vez que cambia la plantilla del controlador de replicación.
- ④ Un desencadenador de cambio de imagen genera la creación de una nueva implementación cada vez que una nueva versión de la imagen **myapp: latest** está disponible en el repositorio.
- ⑤ Define la imagen del contenedor que se implementará: **myapp: latest**.
- ⑥ Especifica los puertos del contenedor.

El último ítem es el servicio, ya analizado en capítulos anteriores:

```

...output omitted...
{
  "kind": "Service",
  "apiVersion": "v1",
  "metadata": {
    "name": "myapp",
    "creationTimestamp": null,
    "labels": {
      "app": "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": {
    "app": "myapp",
    "deploymentconfig": "myapp"
}
},
"status": {
    "loadBalancer": {}
}
}
```

**nota**

De manera predeterminada, el comando **oc new-app** no crea una ruta. Puede crear una ruta después de la creación de la aplicación. Sin embargo, una ruta se crea automáticamente al usar la consola web porque usa una plantilla.

Después de crear una nueva aplicación, comienza el proceso de compilación. Use el comando **oc get builds** para ver una lista de compilaciones de aplicaciones:

\$ oc get builds					
NAME	TYPE	FROM	STATUS	STARTED	DURATION
php-helloworld-1	Source	Git@9e17db8	Running	13 seconds ago	

OpenShift permite ver los registros de compilación. El siguiente comando muestra las últimas líneas del registro de compilación:

```
$ oc logs build/myapp-1
```

**Importante**

Si la compilación aún no se está **ejecutando** o el pod **s2i-build** aún no ha sido implementado por OpenShift, el comando anterior muestra un error. Solo espere un momento y vuelva a intentarlo.

Desencadene un nuevo compilador con el comando **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
```

Relación entre las configuraciones de compilación e implementación

El pod **BuildConfig** debe crear las imágenes en OpenShift y direccionarlas al registro del contenedor interno. Los códigos fuente o las actualizaciones de contenido normalmente necesitan una nueva compilación para garantizar la actualización de la imagen.

El pod **DeploymentConfig** debe implementar pods en OpenShift. El resultado de la ejecución de un pod **DeploymentConfig** es la creación de pods con las imágenes implementadas en el registro del contenedor interno. Todo pod en ejecución existente puede destruirse, según cómo se configure el recurso **DeploymentConfig**.

Los recursos **BuildConfig** y **DeploymentConfig** no interactúan directamente. El recurso **BuildConfig** crea o actualiza una imagen del contenedor. **DeploymentConfig** reacciona a este evento de nueva imagen o imagen actualizada y crea pods a partir de la imagen de contenedor.



Referencias

Compilación de fuente a imagen (S2I)

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/builds/build-strategies#build-strategy-s2i_build-strategies

Repositorio S2I de GitHub

<https://github.com/openshift/source-to-image>

► Ejercicio Guiado

Creación de una aplicación en contenedores con fuente a imagen

En este ejercicio, explorará un contenedor de fuente a imagen, compilará una aplicación a partir de código fuente e implementará la aplicación en un clúster de OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Describir el diseño de un contenedor de Fuente a imagen (Source-to-Image) y los scripts usados para compilar y ejecutar una aplicación dentro del contenedor.
- Compilar una aplicación a partir de código fuente con la interfaz de la línea de comandos de OpenShift.
- Verificar la implementación correcta de la aplicación con la interfaz de la línea de comandos de OpenShift.

Andes De Comenzar

Ejecute el siguiente comando para descargar los archivos relevantes del trabajo de laboratorio y configurar el entorno:

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

- 1. Examine el código fuente para el contenedor de Fuente a imagen (Source-to-Image) de la versión 5.6 de PHP.
- 1.1. Vaya al directorio del trabajo de laboratorio.

```
[student@workstation ~]$ cd ~/D0180/labs/openshift-s2i
```

- 1.2. Use el comando **tree** para revisar los archivos que componen la imagen de contenedor.

```
[student@workstation openshift-s2i]$ tree s2i-php-container
s2i-php-container/
└── 5.6
    ├── cccp.yml
    ├── contrib
    └── etc
        ├── conf.d
        │   ├── 00-documentroot.conf.template
        │   └── 50-mpm-tuning.conf.template
        ├── httpdconf.sed
        └── php.d
```

```

|- 10-opcache.ini.template
|- php.ini.template
|- scl_enable
|- Dockerfile
|- Dockerfile.rhel7
|- README.md
|- s2i
|   |- bin
|   |   |- assemble
|   |   |- run
|   |   |- usage
|   |- test
|   |   |- run
|   |   |- test-app
|   |       |- composer.json
|   |       |- index.php
|- hack
|   |- build.sh
|   |- common.mk
|- LICENSE
|- Makefile
|- README.md

```

- 1.3. Revise el script **s2i-php-container/5.6/s2i/bin/assemble**. Observe cómo mueve el código fuente de PHP desde el directorio **/tmp/src** al directorio de trabajo del contenedor cerca de la parte superior del script. El proceso de fuente a imagen de OpenShift ejecuta el comando **git clone** en el repositorio Git que se provee en la creación que usa el comando **oc new-app** o la consola web. El resto del script soporta la recuperación de paquetes PHP que su aplicación declara como requisitos, en caso de que exista alguno.
 - 1.4. Revise el script **s2i-php-container/5.6/s2i/bin/run**. El contenedor PHP compilado por el proceso de fuente a imagen usa este script como comando predeterminado del contenedor (el equivalente de la instrucción **CMD** en un Dockerfile). Este script es responsable de configurar y ejecutar el servicio HTTP de Apache, que ejecuta el código PHP en respuesta a las solicitudes HTTP.
 - 1.5. Revise el archivo **s2i-php-container/5.6/Dockerfile.rhel7**. Este archivo compila el contenedor base de fuente a imagen de PHP. Instala el servidor HTTP de PHP y Apache desde la librería Red Hat Software Collections Library, copia los scripts de fuente a imagen que examinó en pasos anteriores en su ubicación esperada, y modifica archivos y permisos de archivos según sea necesario para ejecutar en un clúster de OpenShift.
- 2. Inspeccione el código fuente de PHP para la aplicación de muestra y cree y envíe una nueva bifurcación denominada **s2i** para usar durante este ejercicio.
- 2.1. Ingrese su clon local del repositorio **D0180-apps** de Git y extraiga la bifurcación **master** del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation openshift-s2i]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 2.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[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.
```

- 2.3. Revise el código fuente de PHP de la aplicación, en la carpeta **php-helloworld**. Abra el archivo **index.php** en la carpeta **/home/student/D0180-apps/php-helloworld**:

```
<?php
print "Hello, World! php version is " . PHP_VERSION . "\n";
?>
```

La aplicación implementa una respuesta simple que devuelve la versión de PHP en que se ejecuta.

► 3. Prepare el entorno del trabajo de laboratorio.

- 3.1. Cargue la configuración de su entorno de aula.

Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation D0180-apps]$ source /usr/local/etc/ocp4.config
```

- 3.2. Inicie sesión en el clúster 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...
```

- 3.3. Cree un nuevo proyecto que contenga su nombre de usuario de desarrollador de RHOCUP para los recursos que cree durante este ejercicio:

```
[student@workstation D0180-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-s2i
```

► 4. Cree una nueva aplicación PHP con la fuente a imagen del directorio **php-helloworld** con la bifurcación **s2i** que creó en el paso anterior en su bifurcación del repositorio DO180-apps de Git.

- 4.1. Use el comando **oc new-app** para crear la aplicación PHP.

**Importante**

En el siguiente ejemplo, se usa el signo numeral (#) para seleccionar una bifurcación específica desde el repositorio de Git, en este caso, la bifurcación **s2i** creada en el paso anterior.

```
[student@workstation D0180-apps]$ oc new-app --as-deployment-config php:7.3 \
> --name=php-helloworld \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#s2i \
> --context-dir php-helloworld
```

- 4.2. Espere que la compilación se complete y se implemente la aplicación. Verifique que el proceso de compilación comience con el comando **oc get pods**.

```
[student@workstation openshift-s2i]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
php-helloworld-1-build   1/1     Running   0          5s
```

- 4.3. Examine los registros de esta compilación. Use el nombre del pod de compilación para esta compilación, **php-helloworld-1-build**.

```
[student@workstation D0180-apps]$ oc logs --all-containers \
> -f php-helloworld-1-build
Cloning "https://github.com/yourgituser/D0180-apps" ...
Commit: f7cd8963ef353d9173c3a21dcccf402f3616840b (Initial commit, including all
apps previously in course)

...output omitted...

Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-...
php@sha256:f3c9...7546
STEP 1: FROM image-registry.openshift-image-registry.svc:5000/...

...output omitted...

Pushing image ...openshift-image-registry.svc:5000/s2i/php-helloworld:latest...
Getting image source signatures
...output omitted...
STEP 8: RUN /usr/libexec/s2i/assemble
--> Installing application source...
...output omitted...
Copying config sha256:6ce5730f48d9c746e7cbd7ea7b8ed0f15b83932444d1d2bd7711d7...
21.45 KiB / 21.45 KiB 0s
Writing manifest to image destination
Storing signatures
Successfully pushed .../php-helloworld:latest@sha256:63e757a4c0edaeda497dab7...
Push successful
```

Observe el clon del repositorio Git como primer paso de la compilación. A continuación, el proceso de Fuente a imagen (Source-to-Image) crea un nuevo contenedor llamado **s2i/php-helloworld:latest**. El último paso en el proceso de compilación es enviar este contenedor al registro privado de OpenShift.

4.4. Revise **DeploymentConfig** para esta aplicación:

```
[student@workstation D0180-apps]$ oc describe dc/php-helloworld
Name:    php-helloworld
Namespace:      youruser-s2i
Created:        12 minutes ago
Labels:         app=php-helloworld
Annotations:   openshift.io/generated-by=OpenShiftNewApp
Latest Version: 1
Selector:       app=php-helloworld,deploymentconfig=php-helloworld
Replicas:       1
Triggers:       Config, Image(phi-helloworld@latest, auto=true)
Strategy:       Rolling
Template:
  Labels:        app=php-helloworld
                  deploymentconfig=php-helloworld
...output omitted...
  Containers:
    php-helloworld:
      Image:  image-registry.openshift-image-registry.svc:5000/s2i/php-
helloworld@sha256:6d27...b983
      Ports:  8080/TCP, 8443/TCP
      Environment: <none>
      Mounts:  <none>
      Volumes: <none>

  Deployment #1 (latest):
    Name:  php-helloworld-1
    Created: 5 minutes ago
    Status:  Complete
    Replicas: 1 current / 1 desired
    Selector: app=php-helloworld,deployment=php-helloworld-1,deploymentconfig=php-
helloworld
    Labels:  app=php-helloworld,openshift.io/deployment-config.name=php-helloworld
    Pods Status: 1 Running / 0 Waiting / 0 Succeeded / 0 Failed
...output omitted...
```

4.5. Agregue una ruta para probar la aplicación:

```
[student@workstation D0180-apps]$ oc expose service php-helloworld \
> --name ${RHT_OCP4_DEV_USER}-helloworld
route.route.openshift.io/youruser-helloworld exposed
```

4.6. Busque la URL asociada con la nueva ruta:

```
[student@workstation D0180-apps]$ oc get route -o jsonpath='{..spec.host}{"\n"}'
youruser-helloworld-youruser-s2i.wildcard_domain
```

- 4.7. Pruebe la aplicación enviando una solicitud HTTP GET a la URL que obtuvo en el paso anterior:

```
[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 7.3.11
```

- 5. Explore las compilaciones de aplicaciones que se inician con el cambio de la aplicación en su repositorio Git y la ejecución de los comandos adecuados para comenzar una nueva compilación de Fuente a imagen (Source-to-Image).

- 5.1. Ingrese el directorio del código fuente.

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

- 5.2. Edite el archivo **index.php**, como se muestra a continuación:

```
<?php  
print "Hello, World! php version is " . PHP_VERSION . "\n";  
print "A change is a coming!\n";  
?>
```

Guarde el archivo.

- 5.3. Confirme los cambios y envíe el código de vuelta al repositorio Git remoto:

```
[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/youruser/D0180-apps  
 f7cd896..b1324aa s2i -> s2i
```

- 5.4. Inicie un nuevo proceso de compilación de Fuente a image (Source-to-Image) y espere que compile e implemente:

```
[student@workstation php-helloworld]$ oc start-build php-helloworld  
build.build.openshift.io/php-helloworld-2 started  
[student@workstation php-helloworld]$ oc logs php-helloworld-2-build -f  
...output omitted...  
  
Successfully pushed .../php-helloworld:latest@sha256:74e757a4c0edaeda497dab7...  
Push successful
```



nota

Los registros pueden tardar algunos segundos en estar disponibles después de que se inicia la compilación. Si el comando anterior falla, espere unos segundos e inténtelo de nuevo.

Una vez que la segunda compilación haya terminado, use el comando **oc get pods** para verificar que se está ejecutando la nueva versión de la aplicación.

```
[student@workstation php-helloworld]$ oc get pods -w  
...output omitted...  


| NAME                   | READY | STATUS    | RESTARTS | AGE |
|------------------------|-------|-----------|----------|-----|
| php-helloworld-1-build | 0/1   | Completed | 0        | 33m |
| php-helloworld-2-2n70q | 1/1   | Running   | 0        | 1m  |
| php-helloworld-2-build | 0/1   | Completed | 0        | 1m  |


```

Presione **Ctrl+C** para salir del comando **oc get pods -w**.

5.5. Pruebe que la aplicación sirve al nuevo contenido:

```
[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 7.3.11  
A change is a coming!
```

Finalizar

En **workstation**, ejecute el script **lab openshift-s2i finish** para terminar este trabajo de laboratorio.

```
[student@workstation php-helloworld]$ lab openshift-s2i finish
```

Esto concluye el ejercicio guiado.

Creación de aplicaciones con la consola web de OpenShift

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Crear una aplicación con la consola web de OpenShift.
- Administrar y monitorear el ciclo de compilación de una aplicación.
- Examinar recursos para una aplicación.

Acceso a la consola web de OpenShift

La consola web de OpenShift permite al usuario ejecutar muchas de las mismas tareas que el cliente de la línea de comandos de OpenShift. Puede crear proyectos, agregar aplicaciones a proyectos, ver recursos de aplicaciones y manipular configuraciones de aplicaciones según sea necesario. La consola web de OpenShift se ejecuta como uno o más pods, y cada pod se ejecuta en un nodo maestro.

La consola web se ejecuta en un navegador web. La URL predeterminada tiene el formato `https://console-openshift-console.{wildcard DNS domain for the RHOPC cluster}/`. De manera predeterminada, OpenShift genera un certificado autofirmado para la consola web. Debe confiar en este certificado para obtener acceso.

La consola web usa una API REST para comunicarse con el clúster de OpenShift. De forma predeterminada, se accede al extremo de la API REST con un nombre DNS diferente y el certificado autofirmado. También debe aprobar este certificado para el extremo de la API REST.

Una vez que haya aprobado los dos certificados de OpenShift, la consola requiere autenticación para continuar.

Administración de proyectos

Al iniciar sesión correctamente, en **Home (Inicio) → Projects (Proyectos)**, se muestra una lista de proyectos a los que puede acceder. Desde esta página, puede crear, editar o eliminar un proyecto.

Si tiene permiso para ver las métricas del clúster, en su lugar, se le redirige a la página **Home (Inicio) → Overview (Descripción general)**. En esta página, se muestra información general y métricas acerca del clúster. El ítem de menú **Overview (Descripción general)** está oculto para los usuarios sin autoridad para ver las métricas del clúster.

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

Figura 6.9: Página de inicio de la consola web de OpenShift

El ícono de puntos suspensivos al final de cada fila proporciona un menú con las acciones del proyecto. Seleccione la entrada apropiada para editar o eliminar el proyecto.

Si hace clic en el enlace de un proyecto en esta vista, será redirigido a la página **Project Status** (Estado del proyecto), donde se muestran todas las aplicaciones creadas dentro de ese espacio de proyecto.

Navegación por la consola web

Un menú de navegación se encuentra en el lado izquierdo de la consola web. El menú incluye dos perspectivas: **Administrator** (Administrador) y **Developer** (Desarrollador). Cada ítem del menú se expande para proporcionar acceso a un conjunto de funciones de administración relacionadas. Cuando se selecciona la perspectiva **Administrator**, estos ítems son los siguientes:

Inicio

El menú de inicio permite que los usuarios accedan rápidamente a proyectos y recursos. Desde este menú, puede explorar y gestionar proyectos, buscar o explorar los recursos del clúster e inspeccionar los eventos del clúster.

Operadores

OperatorHub es un catálogo que le permite descubrir, buscar e instalar operadores en el clúster. Después de instalar un operador, puede usar la opción **Installed Operators** (Operadores instalados) para administrar el operador o buscar otros operadores instalados.



nota

Las últimas versiones de Kubernetes implementan muchos controladores como **operadores**. Los operadores son componentes de complementos (plug-ins) de Kubernetes que pueden reaccionar ante eventos de clúster y controlar el estado de los recursos. Los operadores y CoreOS Operator Framework están fuera del alcance de este documento.

Cargas de trabajo

Estas opciones permiten la administración de varios tipos de recursos Kubernetes y OpenShift, como pods y configuraciones de implementación. Otras opciones de implementación avanzadas a las que se puede acceder desde este menú, como los mapas de configuración, los secretos y los trabajos cron, están fuera del alcance del curso.

Conexiones en red

Este menú contiene opciones para administrar los recursos de OpenShift que afectan el acceso a la aplicación, como servicios y rutas, para un proyecto. Existen otras opciones disponibles para configurar una política de red OpenShift o Ingress, pero estos temas están fuera del alcance de este curso.

Almacenamiento

Este menú contiene opciones para configurar el almacenamiento persistente para aplicaciones de proyectos. En particular, los volúmenes persistentes y las reclamaciones de volúmenes persistentes para un proyecto se administran desde el menú **Storage (Almacenamiento)**.

Compilaciones

Mediante la opción **Build Configs** (Configuraciones de compilación), se muestra una lista de configuraciones de compilación del proyecto. Haga clic en un enlace de configuración de compilación en esta vista para acceder a una página de descripción general para la configuración de compilación especificada. Desde esta página, puede ver y editar la configuración de compilación de la aplicación.

Mediante la opción **Builds** (Compilación), se proporciona una lista de los procesos de compilación recientes para las imágenes del contenedor de aplicaciones en el proyecto. Haga clic en el enlace de una compilación particular para acceder a los registros de compilación de ese proceso de compilación particular.

Mediante la opción **Image Streams** (Flujo de imágenes), se proporciona una lista de secuencias de imágenes definidas en el proyecto. Haga clic en una entrada de flujo de imágenes en esta lista para acceder a una página de vista general a fin de ver y administrar ese flujo de imágenes.

Cómputo

Proporciona opciones para acceder y administrar los nodos de cómputo del clúster de OpenShift. Desde esta página, también puede configurar el autoescalamiento y las verificaciones de estado de los nodos.

Administración de usuarios

Con esta opción, puede configurar la autenticación y la autorización mediante usuarios, grupos, roles, vinculaciones de roles y cuentas de servicio.

Administration [Administración]

Proporciona opciones para administrar la configuración del clúster y del proyecto, como cuotas de recursos y controles de acceso basados en roles. Las funciones de la sección **Administration** (Administración) están fuera del alcance de este curso.

Creación de nuevas aplicaciones

Desde la perspectiva **Developer** (Desarrollador), use **+Add** (+Añadir) para seleccionar una forma de crear una nueva aplicación en un proyecto de OpenShift. Puede agregar una aplicación desde **Developer Catalog** (Catálogo para desarrolladores), que ofrece una selección de plantillas de fuente a imagen (S2I), imágenes de compilador y gráficos de Helm para crear aplicaciones específicas para la tecnología. Seleccione la plantilla que desee y proporcione la información necesaria para implementar la nueva aplicación.

Usted no está limitado a implementar una aplicación del catálogo. También puede implementar una aplicación usando:

- Una imagen de contenedor alojada en un registro de contenedor remoto
- Un archivo YAML que especifica los recursos de Kubernetes y OpenShift que se crearán
- Una imagen de compilador con el código fuente de su propio repositorio de git.
- Un Dockerfile.

Para crear una aplicación con uno de estos métodos, seleccione la opción adecuada en la página **Add** (Aregar). Use la opción **Container Image** (Imagen de contenedor) para implementar una imagen de contenedor existente. Use la opción **YAML** para crear los recursos especificados en un archivo YAML. Use la opción **From Git** (De Git) para implementar el código fuente mediante una imagen de compilador. Use la opción **From Dockerfile** (De Dockerfile) para especificar y compilar la imagen desde un Dockerfile específico. Las opciones **Databases** (Bases de datos), **Operator Backed** (Respalgado por operador) y **Helm Chart** (Gráfico Helm) son accesos directos al catálogo.

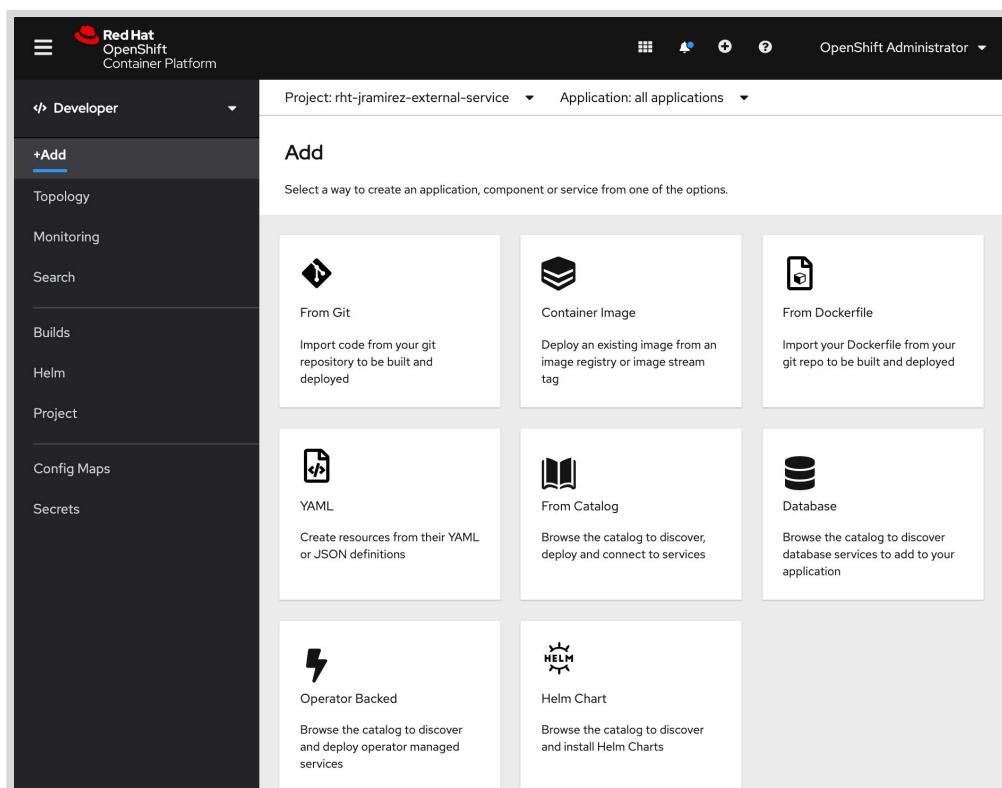


Figura 6.10: Página Catálogo del desarrollador de OpenShift

Administración de compilaciones de una aplicación

Desde la perspectiva **Administrator**, haga clic en la opción **Build Configs** (Configuraciones de compilación) del menú **Builds (Compilaciones)** después de agregar una aplicación de fuente a imagen a un proyecto. Se puede acceder a la nueva configuración de la compilación desde esta vista:

Name	Namespace	Labels	Created
BC todoapp	NS test	app=todoapp	Aug 3, 1:42 pm

Figura 6.11: Página de configuraciones de compilación de OpenShift

Haga clic en una configuración de compilación en la lista para ver una página de descripción general para la configuración de la compilación seleccionada. Desde la página de descripción general, puede realizar las siguientes acciones:

- Ver los parámetros de configuración de la compilación, como la URL del repositorio Git del código fuente.
- Ver y editar las variables de entorno que se establecen en el contenedor del compilador, durante un proceso de compilación de la aplicación.
- Ver una lista de compilaciones de aplicaciones recientes y hacer clic en una compilación seleccionada para acceder a los registros del proceso de compilación.

Administración de aplicaciones implementadas

En el menú **Workloads (Cargas de trabajo)**, se proporciona acceso a las configuraciones de implementación en el proyecto.

Name	Namespace	Status	Labels	Pod Selector
DC todoapp	NS test	1 of 1 pods	app=todoapp, deploymentconfig=todoapp	

Figura 6.12: Menú de cargas de trabajo de OpenShift

Haga clic en una entrada de configuración de implementación en la lista para ver una página de información general para la selección. Desde la página de descripción general, puede realizar las siguientes acciones:

- Ver los parámetros de configuración de implementación, como las especificaciones de una imagen de contenedor de aplicación.
- Cambiar el número deseado de pods de aplicación para escalar manualmente la aplicación.
- Ver y editar las variables de entorno que se establecen en el contenedor de la aplicación implementada.
- Ver una lista de pods de aplicaciones y hacer clic en un pod seleccionado para acceder a los registros de ese pod.

Otras características de la consola web

La consola web le permite:

- Administrar recursos, como cuotas de proyectos, membresías del usuario, secretos y otros recursos avanzados.
- Crear reclamaciones de volúmenes persistentes.
- Monitorear compilaciones, implementaciones, pods y eventos del sistema.
- Crear tuberías de integración e implementación continuas con Jenkins.

El uso detallado de las funciones anteriores está fuera del alcance de este curso.

► Ejercicio Guiado

Creación de una aplicación con la consola web

En este ejercicio, creará, compilará e implementará una aplicación en un clúster de OpenShift con la consola web de OpenShift.

Resultados

Debe ser capaz de crear, compilar e implementar una aplicación en un clúster de OpenShift con la consola web.

Andes De Comenzar

Obtenga los archivos del trabajo de laboratorio mediante la ejecución del script del trabajo de laboratorio:

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

El script del trabajo de laboratorio verifica que se ejecute el clúster de OpenShift.

- ▶ 1. Inspeccione el código fuente de PHP para la aplicación de muestra, y cree y envíe una nueva bifurcación denominada **console** para usar durante este ejercicio.
 - 1.1. Ingrese su clon local del repositorio **D0180-apps** de Git y extraiga la bifurcación **master** del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 1.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[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. Revise el código fuente de PHP de la aplicación, en la carpeta **php-helloworld**. Abra el archivo **index.php** en la carpeta **/home/student/D0180-apps/php-helloworld**:

```
<?php
print "Hello, World! php version is " . PHP_VERSION . "\n";
?>
```

La aplicación implementa una respuesta simple que devuelve la versión de PHP en que se ejecuta.

- 2. Abra un navegador web y navegue a [https://console-openshift-console.\\${RHT_OCP4_WILDCARD_DOMAIN}](https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}) para acceder a la consola web de OpenShift. Inicie sesión y cree un nuevo proyecto con el nombre **youruser-console**.

- 2.1. Cargue la configuración de su entorno de aula.

Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 2.2. Recupere el valor del dominio comodín específico para su clúster mediante **\$RHT_OCP4_WILDCARD_DOMAIN**

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

- 2.3. Abra el navegador Firefox y navegue a [https://console-openshift-console.\\${RHT_OCP4_WILDCARD_DOMAIN}](https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}) para acceder a la consola web de OpenShift. Inicie sesión en la consola de OpenShift con sus credenciales.
- 2.4. Cree un nuevo proyecto con el nombre **youruser-console**. Puede escribir cualquier valor que desee en los otros campos.

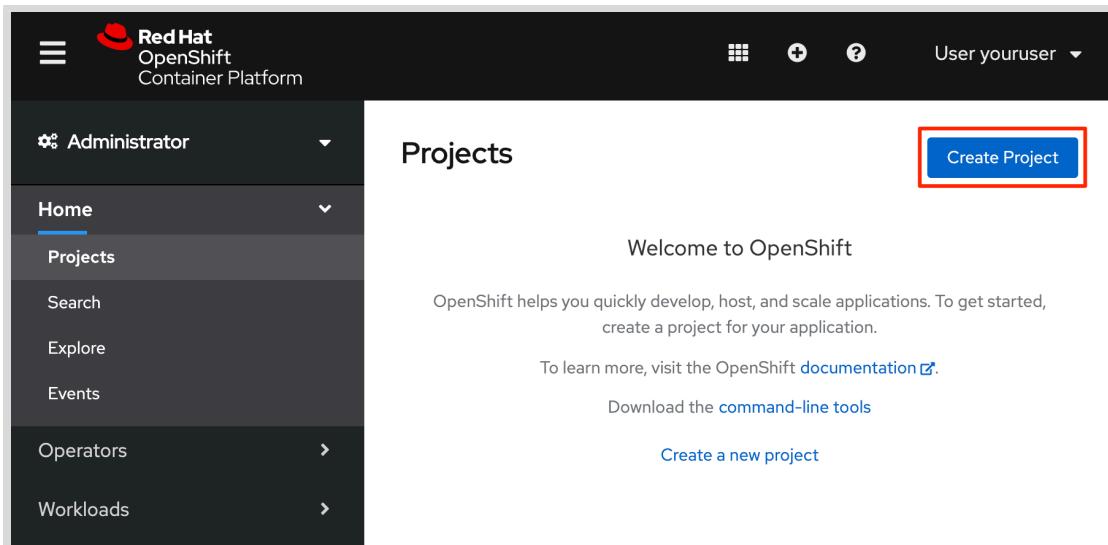


Figura 6.13: Crea un proyecto nuevo.

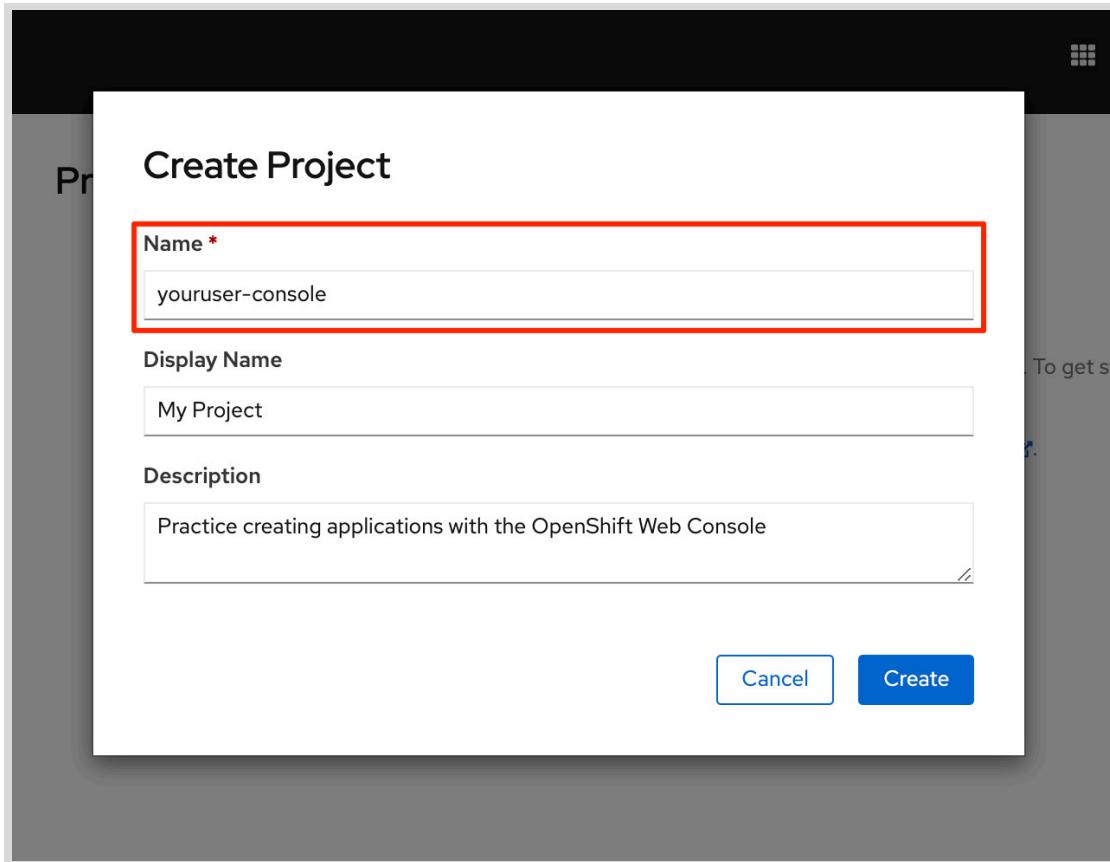


Figura 6.14: Crea un proyecto nuevo.

- 2.5. Una vez que haya completado los campos requeridos, haga clic en **Create** (Crear) en el cuadro de diálogo **Create Project** (Crear proyecto) para ir a la página **Project Status** (Estado del proyecto) para el proyecto **youruser-console**:

Figura 6.15: Página de estado del proyecto

► 3. Cree la aplicación php-helloworld nueva con una plantilla de PHP.

- 3.1. Cambie a la perspectiva de **desarrollador** mediante el menú desplegable que se encuentra en la parte superior del menú de la izquierda:

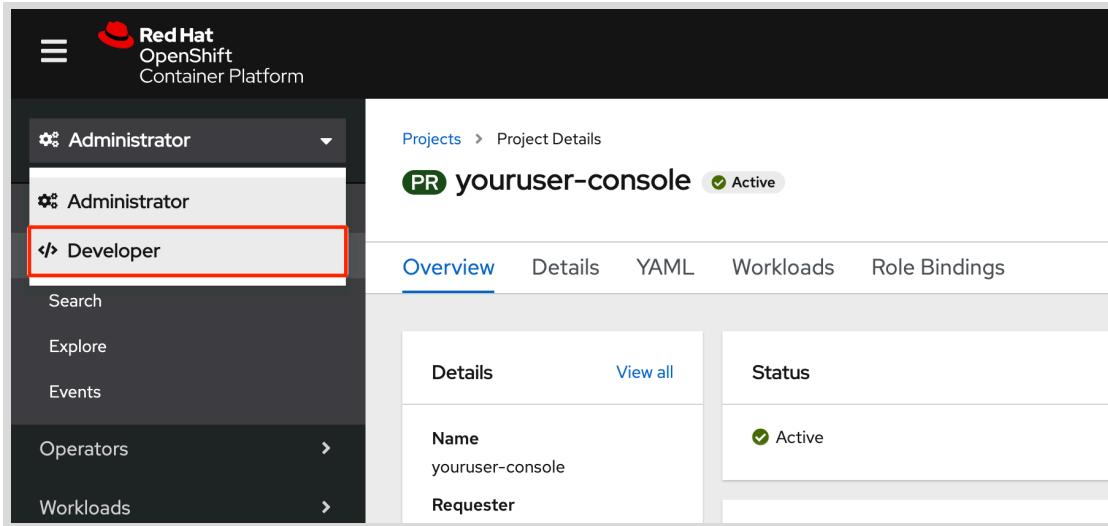


Figura 6.16: Menú desplegable de perspectiva de desarrollador

- 3.2. Haga clic en **From Catalog** (Desde el catálogo) para visualizar una lista de plantillas de tecnología.

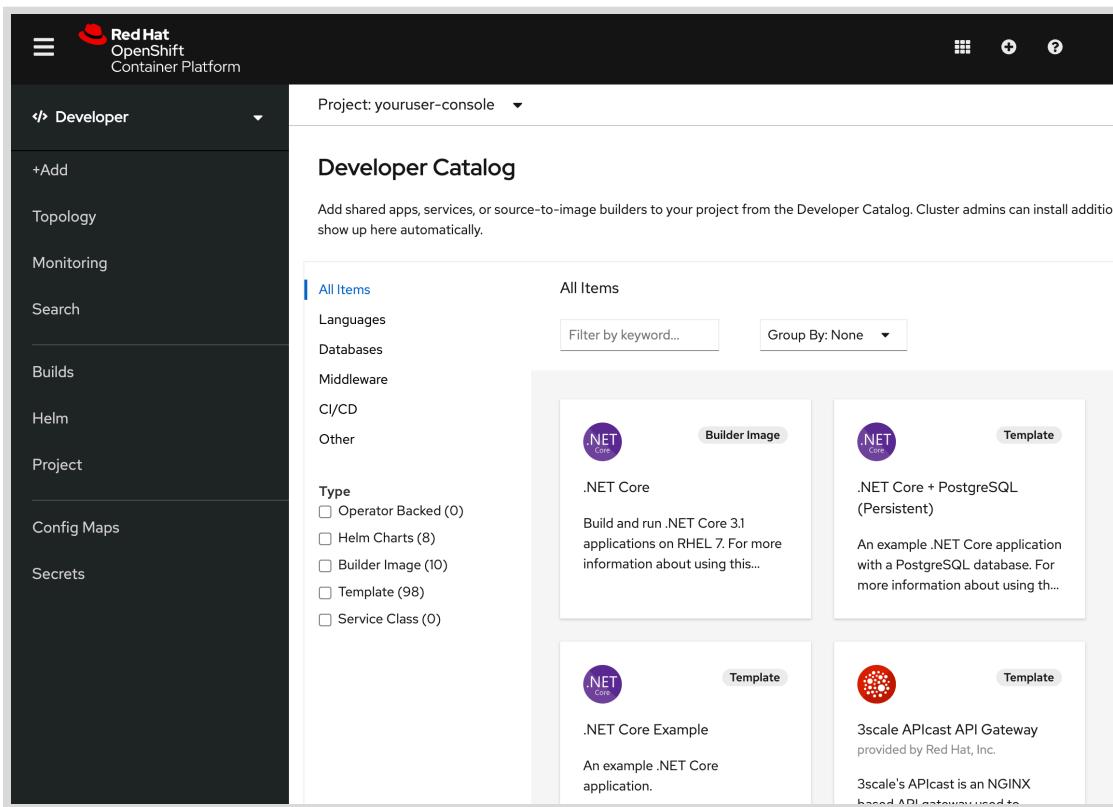


Figura 6.17: Página Catálogo del desarrollador

- 3.3. Ingrese **php** en el campo **Filter by keyword** (Filtrar por palabra clave).

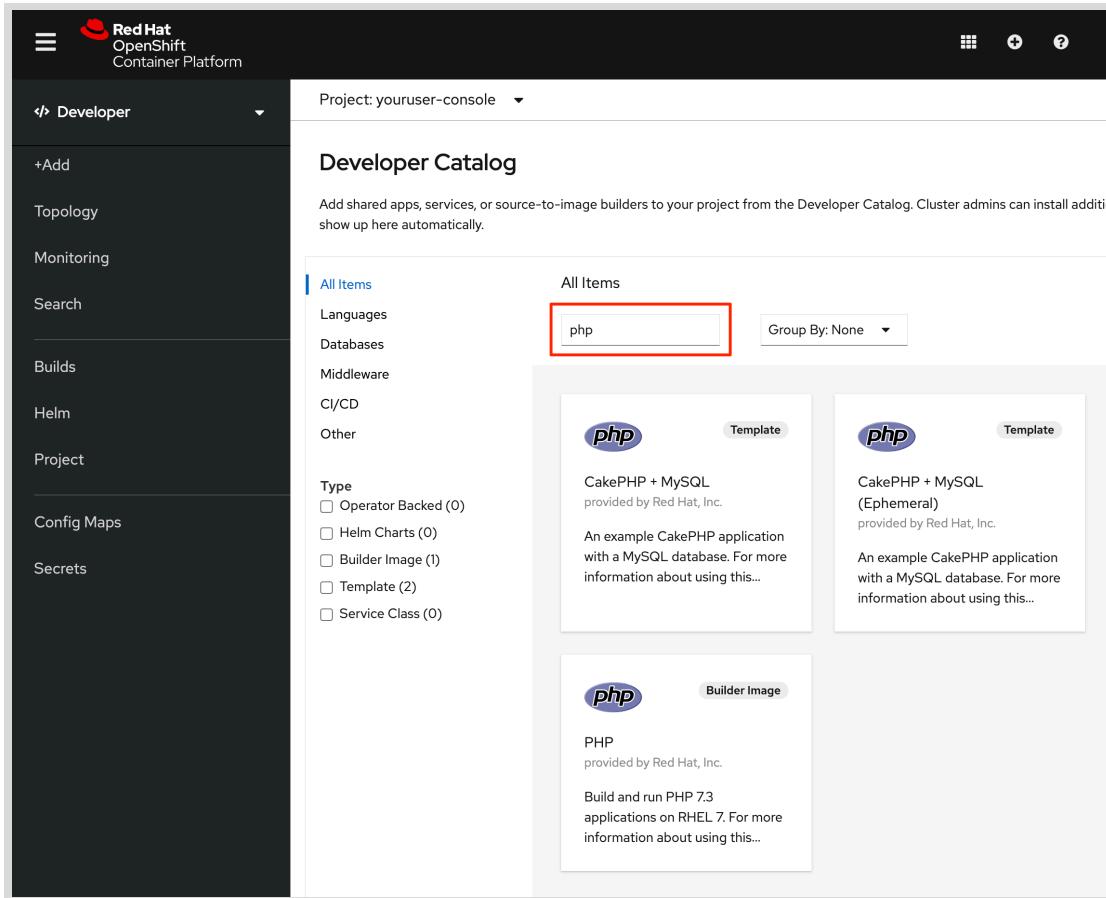


Figura 6.18: Búsqueda de plantillas relacionadas con PHP

- 3.4. Después de filtrar, haga clic en la imagen de compilador de PHP para visualizar el cuadro de diálogo de PHP. Haga clic en **Create Application** (Crear aplicación) para visualizar la página **Create Source-To-Image Application** (Crear aplicación fuente a imagen).

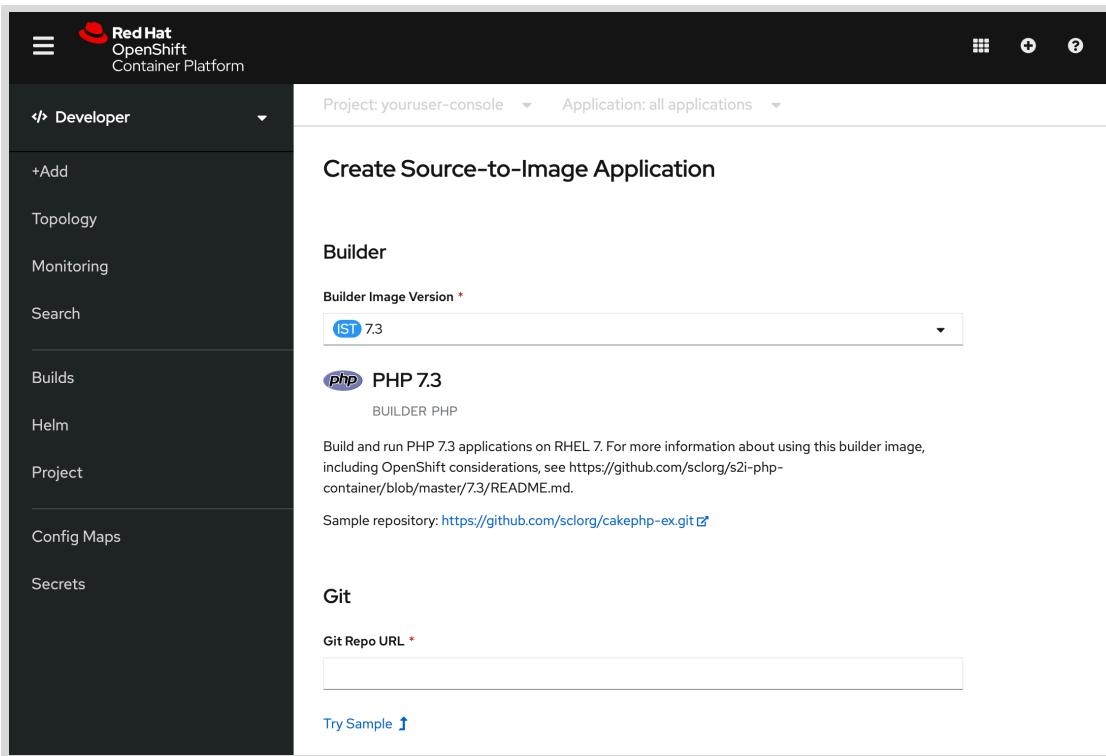


Figura 6.19: Configuración de fuente a imagen para una aplicación PHP

- 3.5. Cambie **Builder Image Version** (Versión de imagen de compilador) a la versión 7.3 de PHP.

Especifique la ubicación del repositorio de Git del código fuente: <https://github.com/yourgituser/D0180-apps>.

Use **Advanced Git Options** (Opciones avanzadas de Git) para establecer el directorio de contexto en **php-helloworld** y la bifurcación **console** para este ejercicio.

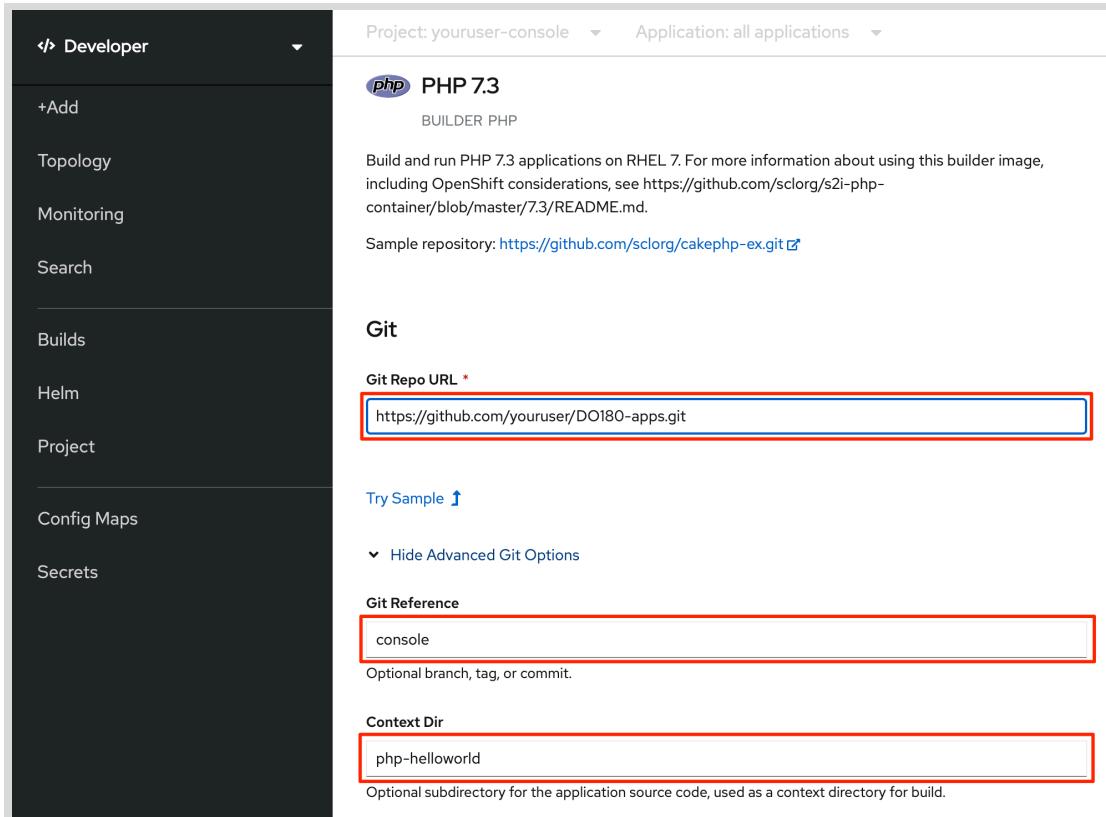


Figura 6.20: Establecimiento de opciones avanzadas de Git para la aplicación

Ingrese **php-helloworld** para el nombre de la aplicación y el nombre usado para los recursos asociados. Seleccione **Deployment Config** (Configuración de implementación) como el tipo de recurso.

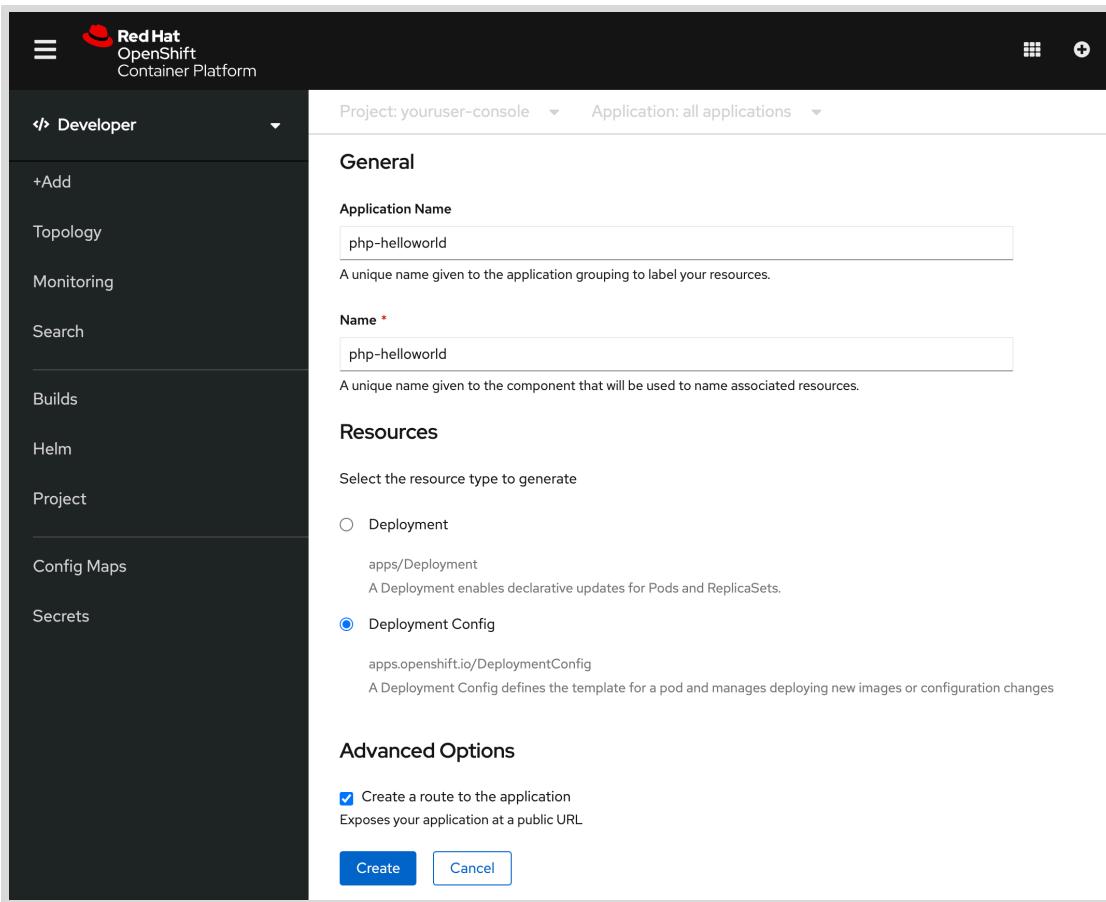


Figura 6.21: Configuración de opciones de la aplicación

Desplácese hasta la parte inferior de la página y seleccione **Create a route to the application** (Crear ruta a la aplicación). Haga clic en **Create** (Crear) para crear los recursos requeridos de OpenShift y Kubernetes para la aplicación.

Será redirigido a la página **Topology** (Topología):

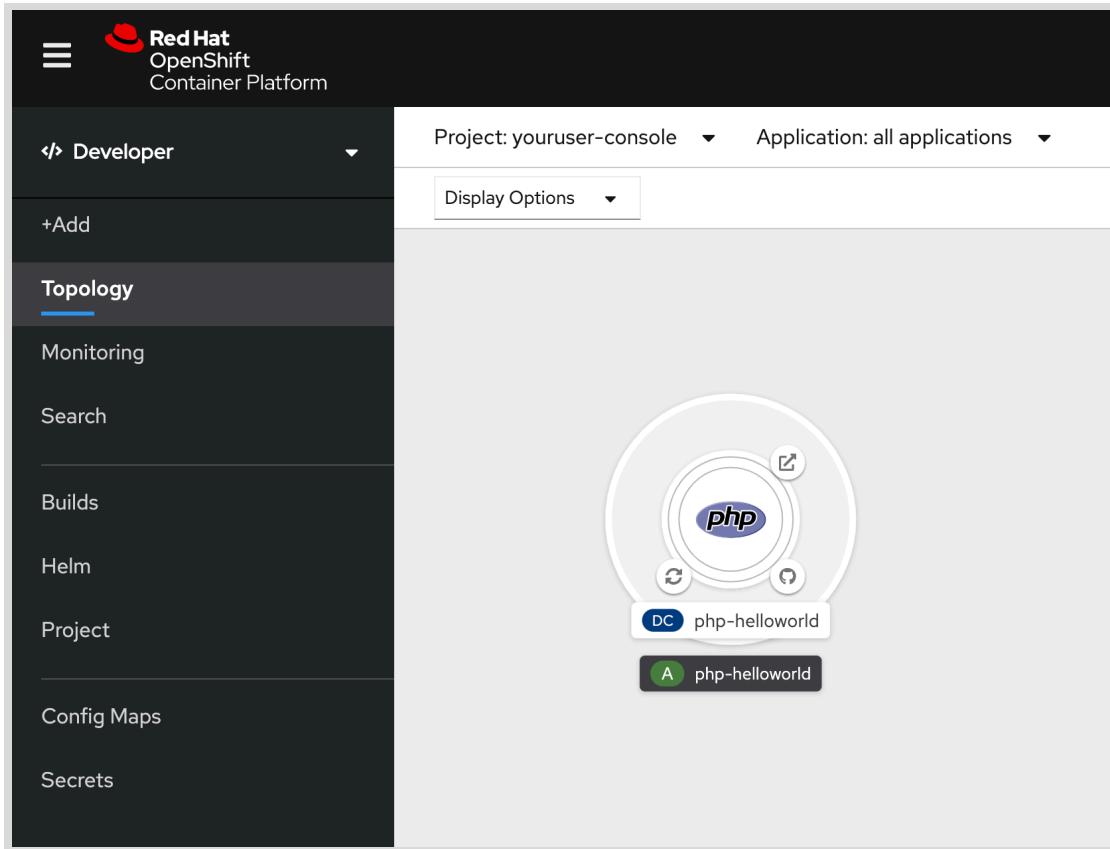


Figura 6.22: Página de topología

En esta página, se indica que se creó la aplicación `php-helloworld`. La anotación **DC** a la izquierda del enlace `php-helloworld` es el acrónimo de **Deployment Config** (Configuración de la implementación). Este enlace lo redirige a una página que contiene información sobre la configuración de implementación de la aplicación.

- 3.6. Vuelva a la perspectiva de **administrador** para el resto del ejercicio:

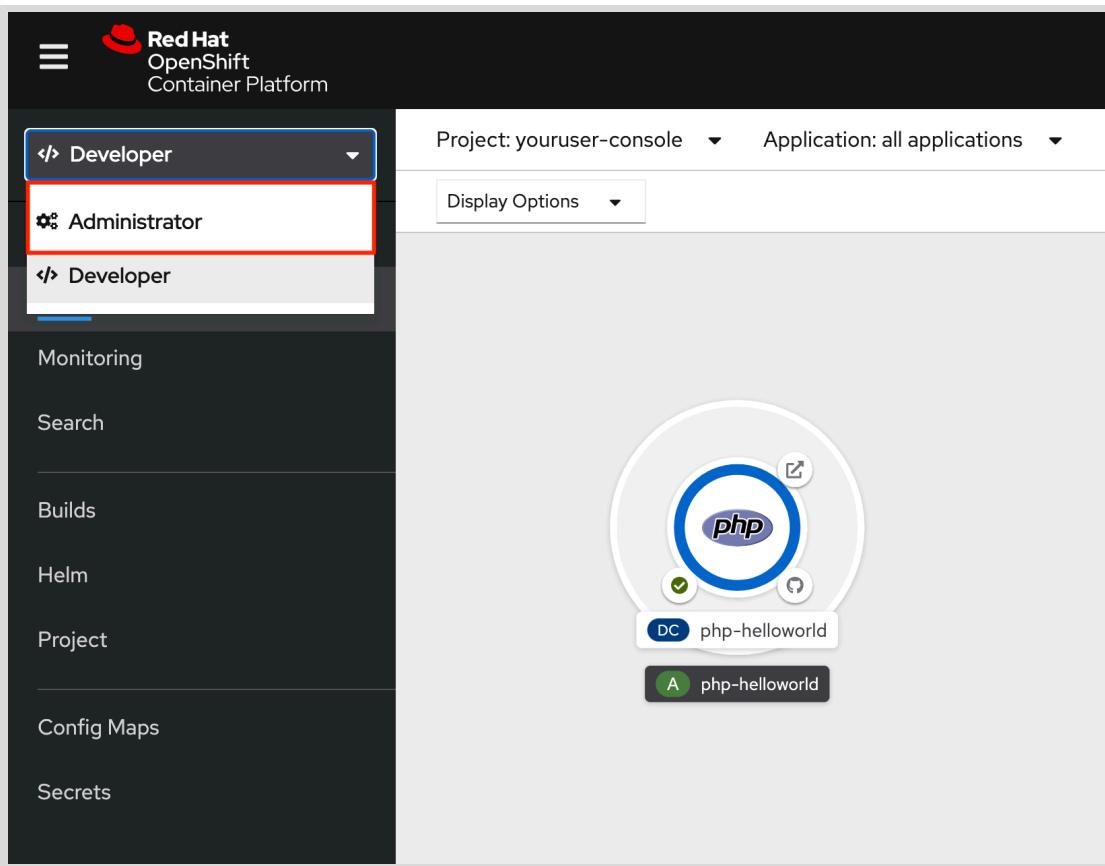


Figura 6.23: Menú desplegable de perspectiva de administrador

- ▶ 4. Use la barra de navegación en el lado izquierdo de la consola web de OpenShift para ubicar información de los recursos de OpenShift y Kubernetes de la aplicación:
 - DeploymentConfig (Configuración de implementación)
 - BuildConfig (Configuración de compilación)
 - Build Logs (Registros de compilación)
 - Servicio (Service)
 - Ruta
- 4.1. Examine la configuración de la implementación. En la barra de navegación, haga clic en **Workloads (Cargas de trabajo)** para mostrar más opciones de menú. Haga clic en **Deployment Configs (Configuraciones de implementación)** para visualizar una lista de configuraciones de implementación para el proyecto **youruser-console**. Haga clic en el enlace **php-helloworld** para visualizar la información de configuración de la implementación.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is dark-themed and includes a navigation bar with 'Administrator', 'Home', 'Projects', 'Search', 'Explore', 'Events', 'Operators', 'Workloads' (selected), 'Pods', 'Deployments', 'Deployment Configs' (selected), 'Stateful Sets', 'Secrets', 'Config Maps', 'Cron Jobs', 'Jobs', 'Daemon Sets', 'Replica Sets', and 'Replication Controllers'. The main content area has a light background. At the top, it says 'Project: youruser-console'. Below that, 'Deployment Configs > Deployment Config Details' is shown. A large blue circle icon contains the number '1 pod'. The 'Details' tab is selected, followed by 'YAML', 'Replication Controllers', 'Pods', 'Environment', and 'Events'. The 'Deployment Config Details' section shows the following information:

Name	Latest Version
php-helloworld	1
Namespace	Message
NS youruser-console	config change
Labels	Update Strategy
app=php-helloworld app.kubernetes.io/component=php-helloworld app.kubernetes.io/instance=php-helloworld app.kubernetes.io/name=php app.kubernetes.io/part-of=php-helloworld app.openshift.io/runtime=php app.openshift.io/runtime-version=7.3	Rolling
Pod Selector	Min Ready Seconds
app=php-helloworld, deploymentconfig=php-helloworld	Not Configured
Triggers	
	ImageChange, ConfigChange

Figura 6.24: Página de descripción general de la configuración de implementación de la aplicación

Explore la información disponible en la pestaña **Details** (Detalles). Es posible que la compilación aún se esté ejecutando cuando llegue a esta página; por lo tanto, es probable que DC no tenga un valor de **1 pod**, aún.

Si hace clic en los iconos de flecha arriba y abajo junto al gráfico de anillos que indica la cantidad de pods, puede escalar (scale up) o reducir (scale down) la aplicación horizontalmente.

- 4.2. Examine la configuración de la compilación. En la barra de navegación, haga clic en **Builds (Compilaciones)** para mostrar más opciones de menú. Haga clic en **Build Configs (Configuraciones de compilación)** para visualizar una lista de configuraciones de compilación para el proyecto **youruser-console**. Haga clic en el enlace **php-helloworld** para visualizar la configuración de la compilación de la aplicación.

Figura 6.25: Página de descripción general de la configuración de la compilación de la aplicación

Explore la información disponible en la pestaña **Details** (Detalles). Mediante la pestaña **YAML**, puede ver y editar la configuración de compilación como un archivo YAML. En la pestaña **Builds** (Compilaciones), se proporciona una lista histórica de compilaciones, junto con un enlace a más información para cada compilación. En la pestaña **Environment** (Entorno), puede ver y editar variables de entorno para el entorno de compilación de la aplicación. En la pestaña **Events** (Eventos), se muestra una lista de eventos y metadatos relacionados con la compilación.

- 4.3. Examine los registros para la compilación de fuente a imagen de la aplicación. En el menú **Builds (Compilaciones)**, haga clic en **Builds (Compilaciones)** para visualizar una lista de las compilaciones recientes para el proyecto **youruser-console**.

Haga clic en el enlace **php-helloworld-1** para acceder a la información de la primera compilación de la aplicación **php-helloworld**:

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, a sidebar menu is open under the 'Administrator' section, with 'Builds' selected. The main content area shows a build named 'php-helloworld-1' which is 'Complete'. The 'Details' tab is selected, displaying memory usage (100 MB), CPU usage (100m), and filesystem usage (0 B) over time. Below these charts, detailed build information is provided: Name (php-helloworld-1), Status (Complete), Namespace (youruser-console), Type (Source), Labels (app=php-helloworld, app.kubernetes.io/part-of=console, app.kubernetes.io/instance=php-helloworld), and Git Repository (https://github.com/yourgithubuser/DO180-apps.git).

Figura 6.26: Página de descripción general de la compilación de una aplicación

Explore la información disponible en la pestaña **Details** (Detalles). A continuación, haga clic en la pestaña **Logs** (Registros). Un cuadro de texto desplazable contiene la salida del proceso de compilación:

The screenshot shows the Red Hat OpenShift Container Platform interface with the 'Logs' tab selected for the 'php-helloworld-1' build. The log output is as follows:

```

Log stream ended.
59 lines
Writing manifest to image destination
Storing signatures
--> 5cb438a2c45
5cb438a2c45f0b7620779e8ca5b10b9c7e8f3279718686d067dd45fd9dec5de
Pushing image image-registry.openshift-image-registry.svc:5000/youruser-console/php-helloworld:latest ...
Getting image source signatures
Copying blob sha256:65b3f0244f1283afc60b050b38243d16fba6725550a6abbff9eb5b67ba59b461
Copying blob sha256:fcb206e9329a1674dd9e8fbfe45c9be28d0d0cbabba3c6bb67a2f2fcf2a
Copying blob sha256:9f04ed9c572e5de5b715b5f831e573d1ad7080933982a25113c443eb6bd9497e
Copying blob sha256:c9ae4966e1018b89a5d0bd37ef429bb5a893e52cff040a40e325be5117620c1
Copying blob sha256:e7021e0589e97471d99c4265b7c8e64da328e40f116b5f260353b2e0a2adb373
Copying blob sha256:9e7a6dc796f0a75c560158a9f9e30fb8b5a90cb53edce9ffbf5f778406e4de39
Copying config sha256:5cb438a2c45f80b7620779e8ca5b10b9c7e8f3279718686d067dd45fd9dec5de
Writing manifest to image destination
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/youruser-console/php-helloworld@sha256:5cb438a2c45f80b7620779e8ca5b10b9c7e8f3279718686d067dd45fd9dec5de
Push successful
  
```

Figura 6.27: Registros para la compilación de una aplicación

Cuando Podman compila una imagen de contenedor, se observa una salida similar en comparación con la que se muestra en el navegador.

- 4.4. Localice información para el servicio de la aplicación **php-helloworld**. En la barra de navegación, haga clic en **Networking (Redes)** para revelar más opciones de menú. Haga clic en **Services (Servicios)** para visualizar una lista de servicios para el proyecto **youruser-console**. Haga clic en el enlace **php-helloworld** para visualizar la información asociada con el servicio de la aplicación:

Figura 6.28: Página de descripción general del servicio

Explore la información disponible en la pestaña **Details** (Detalles). Mediante la pestaña **YAML**, puede ver y editar la configuración del servicio como un archivo YAML. En la pestaña **Pods**, se muestra la lista actual de pods que proporcionan el servicio de la aplicación.

- 4.5. Localice la información de la ruta externa para la aplicación. En la barra de navegación, haga clic en **Networking → Routes** (Redes > Rutas) para visualizar una lista de rutas configuradas para el proyecto **youruser-console**. Haga clic en el enlace **php-helloworld** para visualizar la información asociada con la ruta de la aplicación:

The screenshot shows the Red Hat OpenShift web interface. The left sidebar is titled 'Administrator' and includes links for Home, Projects, Search, Explore, Events, Operators, Workloads, Networking (selected), Services, Routes, Ingresses, Network Policies, Storage, and Builds. The main content area is titled 'Project: youruser-console'. It shows a 'Routes' section with one entry: 'RT php-helloworld' (Accepted). The 'Details' tab is selected, displaying route information: Name (php-helloworld), Location (http://php-helloworld-youruser-console.apps.cluster.lab.example.com), Namespace (youruser-console), Status (Accepted), Labels (app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php, app.kubernetes.io/part-of=console, app.openshift.io/runtime=php, app.openshift.io/runtime-version=7.3), Path (-), and Router Canonical Hostname (apps.cluster.lab.example.com).

Figura 6.29: Página de descripción general de la ruta

Explore la información disponible en la pestaña **Details** (Detalles). En el campo **LOCATION** (UBICACIÓN), se proporciona un enlace a la ruta externa para la aplicación; `http://php-helloworld-${RHT_OCP4_DEV_USER}-console.${RHT_OCP4_WILDCARD_DOMAIN}`. Haga clic en el enlace para acceder a la aplicación en una nueva pestaña:

The screenshot shows a Mozilla Firefox browser window. The address bar displays 'php-helloworld-console.apps.cluster.lab.example.com'. The page content shows the text 'Hello, World! php version is 7.0.27'.

Figura 6.30: Resultados iniciales de la aplicación PHP

- 5. Modifique el código de la aplicación, confirme el cambio, envíe el código al repositorio Git remoto y desencadene una nueva compilación de la aplicación.

- 5.1. Ingrese en el directorio del código fuente:

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

- 5.2. Agregue el segundo enunciado de la línea de impresión en la página `index.php` para que lea "A change is in the air!" (Hay un cambio en camino) y guarde el archivo. Agregue el cambio al índice Git, confirme el cambio y envíe los cambios al repositorio Git remoto.

```
[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. Desencadene una compilación de la aplicación de forma manual desde la consola web.

En la barra de navegación, haga clic en **Builds → Build Configs** (Compilaciones > Configuraciones de compilación) y, luego, haga clic en el enlace **php-helloworld** para acceder a la página Build Config Overview (Descripción general de la configuración de la compilación). Desde el menú **Actions (Acciones)** en la parte superior derecha de la pantalla, haga clic en **Start Build** (Iniciar compilación):

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is a navigation menu with sections like Home, Operators, Workloads, Networking, Storage, and Builds. Under Builds, there are sub-options for Build Configs, Builds, and Image Streams. The main content area is titled 'Build Config Details' for a build configuration named 'php-helloworld'. It shows details such as Name (php-helloworld), Type (Source), Namespace (NS youruser-console), Git Repository (https://github.com/yourgithubuser/DO180-apps.git), Labels (app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php, app.kubernetes.io/part-of=console, app.openshift.io/runtime=php, app.openshift.io/runtime-version=7.3), Git Ref (console), Context Dir (php-helloworld), and Build From (IST php7.3). To the right of the main content, there is a 'Actions' dropdown menu with options: Start Build, Edit Labels, Edit Annotations, Edit Build Config, and Delete Build Config. The 'Start Build' option is highlighted with a red box.

Figura 6.31: Iniciar una compilación de la aplicación

Será redirigido a la página Build Overview (Descripción general de la compilación) de la nueva compilación. Haga clic en la pestaña **Logs** (Registros) para monitorear el progreso de la compilación. La última línea de una compilación correcta contiene **Push successful** (Push correcto).

Cuando se complete la compilación, se iniciará la implementación. Vaya a la sección **Workloads → Pods** (Cargas de trabajo > Pods) y espere que el nuevo pod esté implementado y en ejecución.

- 5.4. Vuelva a cargar la URL `http://php-helloworld-${RHT_OCP4_DEV_USER}-console.${RHT_OCP4_WILDCARD_DOMAIN}` en el navegador. La respuesta de la aplicación corresponde al código fuente actualizado:

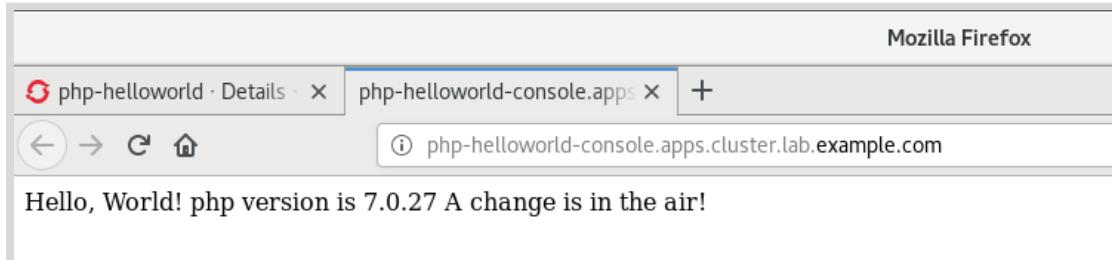


Figura 6.32: Salida de la aplicación web actualizada

- 6. Elimine el proyecto. En la barra de navegación, haga clic en **Home → Projects** (Inicio > Proyectos). Haga clic en el icono en el lado derecho de la fila que contiene una entrada para el proyecto **youruser-console**. Haga clic en **Delete Project** (Eliminar proyecto) en el menú que aparece.

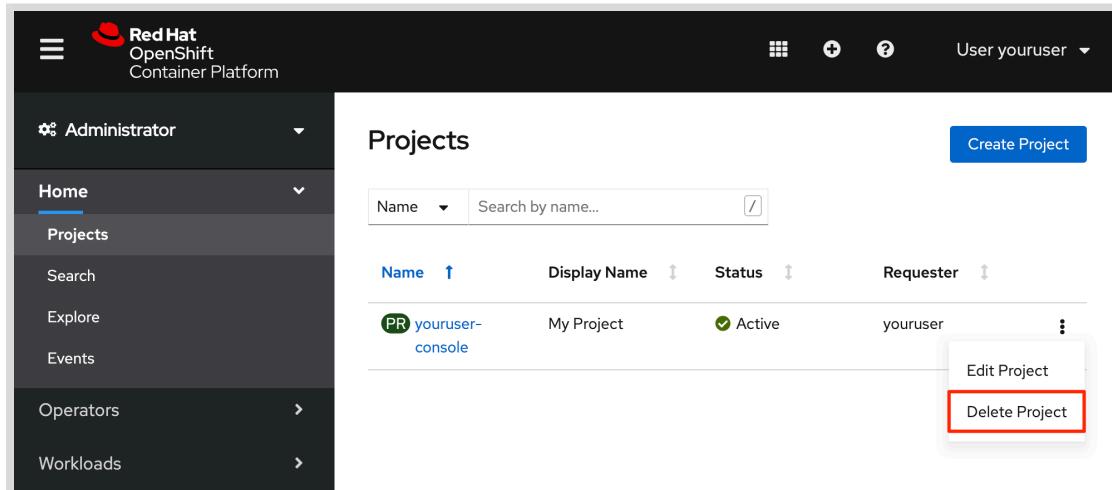


Figura 6.33: Eliminar un proyecto

Ingrese **youruser-console** en el cuadro de diálogo de confirmación y haga clic en **Delete** (Eliminar).

Finalizar

En **workstation**, ejecute el script **lab openshift-webconsole finish** para terminar este trabajo de laboratorio.

```
[student@workstation php-helloworld]$ lab openshift-webconsole finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Implementación de aplicaciones contenedorizadas en OpenShift

Lista de verificación de rendimiento

En este trabajo de laboratorio, creará una aplicación con la utilidad Fuente a imagen (Source-to-Image) de OpenShift.

Resultados

Debería poder crear una aplicación de OpenShift y acceder a esta a través de un navegador web.

Andes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Prepare el entorno del trabajo de laboratorio.
 - 1.1. Cargue la configuración de su entorno de aula.
Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

 - 1.2. Inicie sesión en el clúster OpenShift.
 - 1.3. Cree un nuevo proyecto con el nombre "youruser-ocp" para los recursos que cree durante este ejercicio:
2. Cree una aplicación de conversión de temperatura escrita en PHP con la etiqueta del flujo de imágenes **php:7.3**. El código fuente se encuentra en el repositorio de Git en <https://github.com/RedHatTraining/D0180-apps/> en el directorio **temps**. Puede usar la interfaz de la línea de comandos de OpenShift o la consola web para crear la aplicación.
Exponga el servicio de la aplicación para que la aplicación esté accesible desde un navegador web.
3. Verifique que pueda acceder a la aplicación en un navegador web en <http://temps-youruser-ocp.apps.cluster.lab.example.com>.

Evaluación

En **workstation**, ejecute el comando **lab openshift-review grade** para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En **workstation**, ejecute el comando **lab openshift-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

► Solución

Implementación de aplicaciones contenedorizadas en OpenShift

Lista de verificación de rendimiento

En este trabajo de laboratorio, creará una aplicación con la utilidad Fuente a imagen (Source-to-Image) de OpenShift.

Resultados

Debería poder crear una aplicación de OpenShift y acceder a esta a través de un navegador web.

Andes De Comenzar

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Prepare el entorno del trabajo de laboratorio.

- 1.1. Cargue la configuración de su entorno de aula.

Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- 1.2. Inicie sesión en el clúster 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. Cree un nuevo proyecto con el nombre "youruser-ocp" para los recursos que cree durante este ejercicio:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-ocp
```

2. Cree una aplicación de conversión de temperatura escrita en PHP con la etiqueta del flujo de imágenes **php:7.3**. El código fuente se encuentra en el repositorio de Git en <https://github.com/RedHatTraining/D0180-apps/> en el directorio **temps**. Puede usar la interfaz de la línea de comandos de OpenShift o la consola web para crear la aplicación.

Exponga el servicio de la aplicación para que la aplicación esté accesible desde un navegador web.

2.1. Si usa la interfaz de línea de comandos, ejecute los siguientes comandos:

```
[student@workstation ~]$ oc new-app --as-deployment-config \
> php:7.3-https://github.com/RedHatTraining/D0180-apps \
> --context-dir temps --name temps
--> Found image fbe3911 (2 weeks 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
deploymentconfig.apps.openshift.io "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. Monitoree el progreso de la compilación.

```
[student@workstation ~]$ oc logs -f bc/temps
Cloning "http://services.lab.example.com/temps" ...
Commit: 350b6ca43ff05d1c395a658083f74a92c53fc7e9 (Establish remote repository)
Author: your_user <your@email.com>
Date: Tue Jun 26 21:59:41 2018 +0000
...output omitted...
Successfully pushed image-registry.openshift-image-registry.svc:5000/youruser-ocp/
temps@sha256:2c0c...ef8f
Push successful
```

2.3. Verifique que la aplicación se haya implementado.

```
[student@workstation ~]$ oc get pods -w
NAME        READY   STATUS    RESTARTS   AGE
temps-1-build  0/1     Completed   0          5m
temps-1-h76lt  1/1     Running    0          5m
```

Presione **Ctrl+C** para salir del comando **oc get pods -w**.

2.4. Exponga el servicio **temps** para crear una ruta externa para la aplicación.

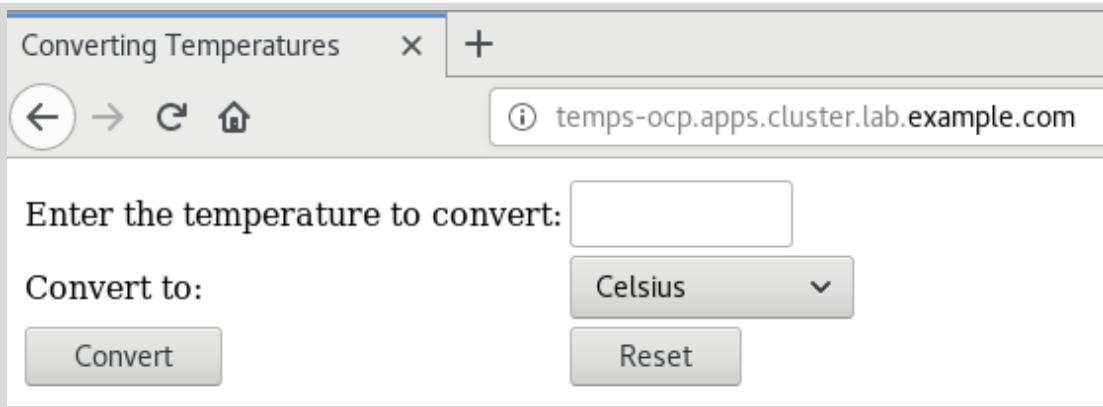
```
[student@workstation ~]$ oc expose svc/temps
route.route.openshift.io/temps exposed
```

3. Verifique que pueda acceder a la aplicación en un navegador web en `http://temps-youruser-ocp.apps.cluster.lab.example.com`.

- 3.1. Determine la URL para la ruta.

```
[student@workstation ~]$ oc get route/temps  
NAME      HOST/PORT      ...  
temps     youruser-ocp.wildcard_comain ...
```

- 3.2. Abra Firefox y navegue a `http://temps-youruser-ocp.wildcard_comain` para verificar que funcione la aplicación del conversor de temperatura.



Evaluación

En `workstation`, ejecute el comando `lab openshift-review grade` para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

En `workstation`, ejecute el comando `lab openshift-review finish` para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- OpenShift Container Platform almacena las definiciones de cada instancia de recurso OpenShift o Kubernetes como un objeto en el servicio de base de datos distribuida del clúster, **etcd**. Los tipos de recursos más comunes son los siguientes: **pod**, **volumen persistente** (PV), **reclamación de volumen persistente** (PVC), **servicio** (SVC), **ruta**, **configuración de implementación** (DC) y **configuración de compilación** (BC).
- Usar el cliente de la línea de comandos de OpenShift **oc** para:
 - Crear, modificar y eliminar proyectos.
 - Crear recursos de aplicaciones dentro de un proyecto.
 - Eliminar, inspeccionar, editar y exportar recursos dentro de un proyecto.
 - Comprobar registros desde pods de aplicación, implementaciones y operaciones de compilación.
- El comando **oc new-app** puede crear pods de aplicación de muchas maneras diferentes: a partir de una imagen de contenedor existente alojada en un registro de imágenes, desde Dockerfiles y desde el código fuente mediante el proceso de fuente a imagen (S2I).
- Fuente a imagen (S2I) es una herramienta que facilita la compilación de una imagen de contenedor desde un código fuente de la aplicación. Esta herramienta recupera el código fuente de una aplicación desde un repositorio Git, inserta el código fuente en un contenedor seleccionado según un lenguaje o una tecnología deseados, y produce una nueva imagen de contenedor que ejecuta la aplicación ensamblada.
- Una **ruta** conecta una dirección IP de acceso público y un nombre de host de DNS a una IP de servicio de acceso interno. Si bien los servicios permiten el acceso de red entre pods dentro de una instancia de OpenShift, las rutas permiten el acceso de red a pods desde los usuarios y las aplicaciones fuera de la instancia de OpenShift.
- Puede crear, compilar, implementar y monitorear aplicaciones con la consola web de OpenShift.

capítulo 7

Implementación de aplicaciones con múltiples contenedores

Meta

Implementar aplicaciones que estén en contenedores con varias imágenes de contenedores.

Objetivos

- Describir las consideraciones para organizar las aplicaciones en contenedores con varias imágenes de contenedores.
- Implementar una aplicación de varios contenedores en OpenShift usando una plantilla.

Secciones

- Consideraciones para aplicaciones con varios contenedores (y ejercicio guiado)
- Implementación de una aplicación con varios contenedores en OpenShift (y ejercicio guiado)

Trabajo de laboratorio

- Implementación de aplicaciones con múltiples contenedores

Consideraciones para aplicaciones con varios contenedores

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de realizar lo siguiente:

- Describir las consideraciones para organizar las aplicaciones en contenedores con varias imágenes de contenedores.
- Aprovechar los conceptos de redes en contenedores.
- Crear una aplicación con múltiples contenedores con Podman.
- Describir la arquitectura de la aplicación To Do List (Lista de tareas).

Aprovechar las aplicaciones con múltiples contenedores

Los ejemplos que se mostraron hasta ahora en este curso han funcionado bien con un solo contenedor. Sin embargo, una aplicación más compleja puede obtener los beneficios de implementar diferentes componentes en diferentes contenedores. Considere una aplicación compuesta de una aplicación web front-end, un back-end de REST y un servidor de base de datos. Esos componentes pueden tener diferentes dependencias, requisitos y ciclos de vida.

Aunque es posible orquestar los contenedores de aplicaciones con múltiples contenedores manualmente, Kubernetes y OpenShift proporcionan herramientas para facilitar la orquestación. Intentar gestionar de forma manual docenas o cientos de contenedores se vuelve complicado rápidamente. En esta sección, volveremos a usar Podman para crear una aplicación simple con múltiples contenedores para demostrar los pasos manuales básicos para la orquestación de contenedores. En las secciones posteriores, usará Kubernetes y OpenShift para orquestar estos mismos contenedores de aplicaciones.

Descubrimiento de servicios en una aplicación con varios contenedores

Podman usa la *Interfaz de red de contenedor (CNI)* para crear una *red definida por software (SDN)* entre todos los contenedores del host. A menos que se indique lo contrario, CNI asigna una nueva dirección IP a un contenedor cuando se inicia.

Cada contenedor expone todos los puertos en otros contenedores de la misma SDN. De esta manera, los servicios están disponibles inmediatamente dentro de la misma red. Los contenedores exponen puertos en redes externas basándose solo en configuración explícita.

Debido a la naturaleza dinámica de las direcciones IP del contenedor, las aplicaciones no pueden depender de direcciones IP fijas o nombres de host de DNS fijos para comunicarse con servicios de middleware y otros servicios de aplicaciones. Los contenedores con direcciones IP dinámicas pueden ser un problema cuando se trabaja con aplicaciones con múltiples contenedores, ya que cada contenedor debe ser capaz de comunicarse con otros contenedores para usar los servicios de los cuales depende.

Por ejemplo, considere una aplicación compuesta por un contenedor de front-end, un contenedor de back-end y una base de datos. El contenedor de front-end necesita recuperar la dirección

IP del contenedor de back-end. De manera similar, el contenedor de back-end debe recuperar la dirección IP del contenedor de la base de datos. Además, la dirección IP podría cambiar si se reinicia un contenedor, de modo que se necesita un proceso para garantizar que un cambio en la IP desencadene una actualización de los contenedores existentes.

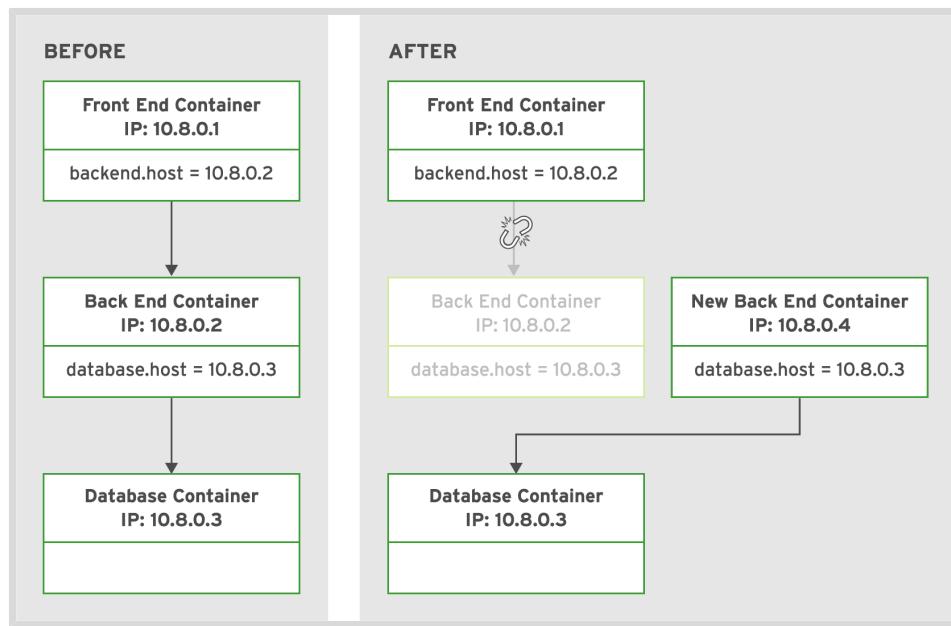


Figura 7.1: Un reinicio rompe enlaces de aplicaciones de tres capas

Tanto Kubernetes como OpenShift proporcionan posibles soluciones al problema de la detectabilidad del servicio y la naturaleza dinámica de las redes del contenedor. Algunas de estas soluciones se tratan más adelante en el capítulo.

Comparación de Podman y Kubernetes

El uso de variables de entorno le permite compartir información entre contenedores con Podman. Sin embargo, aún existen algunas limitaciones y trabajo manual involucrados en garantizar que todas las variables del entorno permanezcan sincronizadas, especialmente cuando trabaja con varios contenedores. Kubernetes proporciona un enfoque para solucionar este problema al crear servicios para sus contenedores, como se vio en capítulos anteriores.

Los pods se conectan con el espacio de nombres de Kubernetes, que OpenShift denomina *proyecto*. Cuando se inicia un pod, Kubernetes agrega automáticamente un conjunto de variables de entorno para cada servicio definido en el mismo espacio de nombres.

Cualquier servicio definido en Kubernetes genera variables de entorno para la dirección IP y el número de puerto donde el servicio está disponible. Kubernetes inserta automáticamente estas variables de entorno en los contenedores desde pods en el mismo espacio de nombres. Estas variables de entorno comúnmente siguen una convención:

- *Mayúscula*: Todas las variables de entorno están configuradas con nombres en mayúscula.
- *Snake case*: Cualquier variable de entorno creada por un servicio por lo general está compuesta de varias palabras separadas por un guion bajo (_).
- *Nombre del servicio primero*: La primera palabra para una variable de entorno creada por un servicio es el nombre del servicio.
- *Tipo de protocolo*: La mayoría de las variables de entornos de red incluyen el tipo de protocolo (TCP o UDP).

Estas son variables de entorno generadas por Kubernetes para un servicio:

- **<SERVICE_NAME>_SERVICE_HOST**: Representa la dirección IP habilitada por un servicio para acceder a un pod.
- **<SERVICE_NAME>_SERVICE_PORT**: Representa el puerto donde se muestra el puerto del servidor.
- **<SERVICE_NAME>_PORT**: Representa la dirección, el puerto y el protocolo provistos por el servicio para acceso externo.
- **<SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>**: Define un alias para **<SERVICE_NAME>_PORT**.
- **<SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PROTO**: Identifica el tipo de protocolo (TCP o UDP).
- **<SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PORT**: Define un alias para **<SERVICE_NAME>_SERVICE_PORT**.
- **<SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_ADDR**: Define un alias para **<SERVICE_NAME>_SERVICE_HOST**.

Por ejemplo, dado el siguiente servicio:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
    name: mysql
spec:
  ports:
    - protocol: TCP
      - port: 3306
  selector:
    name: mysql
```

Las siguientes variables de entorno están disponibles para cada pod creado después del servicio, en el mismo espacio de nombres:

```
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
```

**nota**

Otros nombres de variables de entorno <SERVICE_NAME>_PORT_* relevantes se establecen sobre la base del protocolo. La dirección IP y el número de puerto se establecen en la variable de entorno <SERVICE_NAME>_PORT. Por ejemplo, la entrada **MYSQL_PORT=tcp://10.0.0.11:3306** conduce a la creación de variables de entorno con nombres como **MYSQL_PORT_3306_TCP**, **MYSQL_PORT_3306_TCP_PROTO**, **MYSQL_PORT_3306_TCP_PORT** y **MYSQL_PORT_3306_TCP_ADDR**. Si no se define el componente del protocolo de una variable de entorno, Kubernetes usa el protocolo TCP y asigna los nombres de la variable en consecuencia.

Descripción de la aplicación To Do List (Lista de tareas)

Muchos trabajos de laboratorio de este curso usan una aplicación To Do List (Lista de tareas). Esta aplicación está dividida en tres niveles, como se muestra en la siguiente figura:

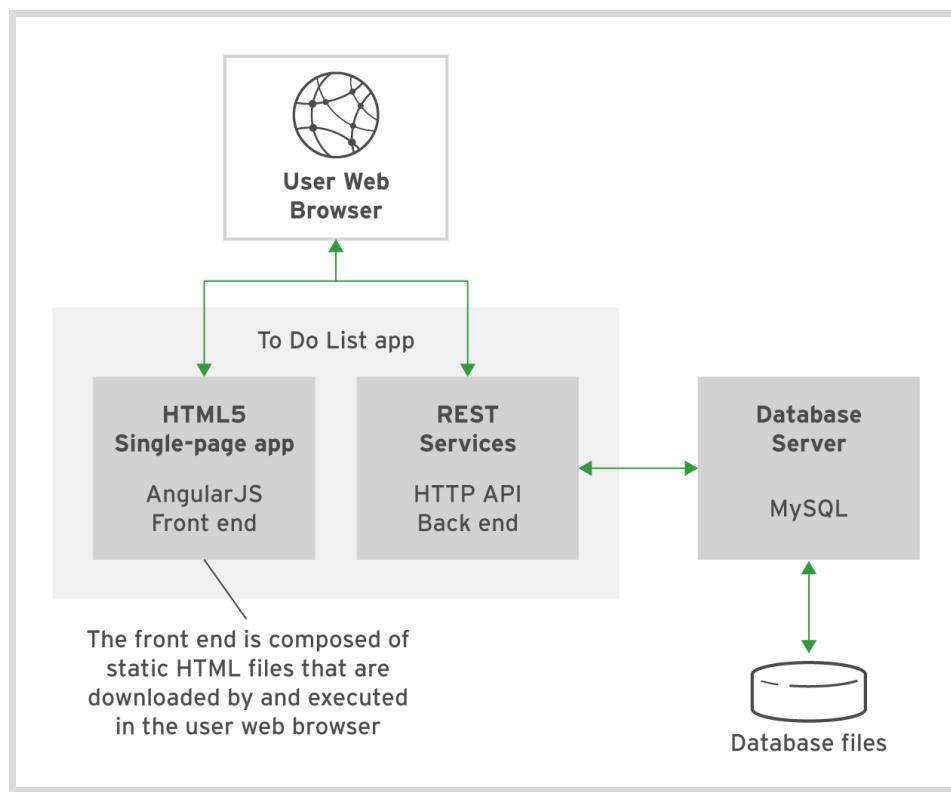


Figura 7.2: Arquitectura lógica de la aplicación To Do List (Lista de tareas)

- El nivel de presentación está diseñado como un front-end HTML5 de una sola página con AngularJS.
- El nivel empresarial está compuesto por un back-end de la API HTTP, con Node.js.
- El nivel de persistencia se basa en un servidor de bases de datos MySQL.

La siguiente figura es una captura de pantalla de la interfaz web de la aplicación:

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

First Previous 1 Next Last

Figura 7.3: Aplicación To Do List

A la izquierda, hay una tabla con ítems para completar y, a la derecha, un formulario para agregar una nueva tarea.

El servidor de registro privado del aula, **services.lab.example.com**, proporciona la aplicación en dos versiones:

nodejs

Representa la manera en que un desarrollador típico crearía la aplicación como una sola unidad, sin interesarse por descomponerla en niveles o servicios.

nodejs_api

Muestra los cambios necesarios para descomponer el nivel empresarial y el nivel de presentación de la aplicación. Cada nivel corresponde a una imagen de contenedor aislada.

Las fuentes de estas versiones de la aplicación están disponibles en la carpeta **todoapp/nodejs** en el repositorio Git en: <https://github.com/RedHatTraining/D0180-apps.git>.

► Ejercicio Guiado

Implementación de la aplicación web y los contenedores MySQL

En este trabajo de laboratorio, creará un script que ejecute y conecte un contenedor de aplicaciones Node.js y el contenedor MySQL.

Resultados

Debería poder vincular contenedores de red para crear una aplicación con varios niveles.

Andes De Comenzar

Debe tener el código fuente de la aplicación To Do List (Lista de tareas) y los archivos del trabajo de laboratorio en **workstation**. Para configurar el entorno para el ejercicio, ejecute el siguiente comando:

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

► 1. Compile la imagen MySQL.

Para este ejercicio, se usa una imagen de MySQL 5.7 personalizada. Se configura para ejecutar automáticamente scripts en el directorio **/var/lib/mysql/init**. Los scripts cargan el esquema y algunos datos de muestra en la base de datos para la aplicación To Do List (Lista de tareas) cuando se inicia un contenedor.

1.1. Revise el Dockerfile.

Con su editor de preferencia, abra y examine el Dockerfile completo, ubicado en **/home/student/D0180/labs/multicontainer-design/images/mysql/Dockerfile**.

1.2. Compile la imagen de la base de datos MySQL.

Para compilar la imagen base, ejecute el siguiente script **podman build**.

```
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-design/images/mysql
[student@workstation mysql]$ sudo podman build -t do180/mysql-57-rhel7 \
> --layers=false .
STEP 1: FROM rhscl/mysql-57-rhel7
Getting image source signatures
...output omitted...
Storing signatures
STEP 2: ADD root /
ERRO[0017] HOSTNAME is not supported for OCI image format, hostname d4f8f8506f2b
will be ignored. Must use `docker` format
--> 41a6...cc7e
STEP 3: COMMIT do180/mysql-57-rhel7
...output omitted...
```

```
Writing manifest to image destination
Storing signatures
--> 8dc1...6933
```

El error que se muestra es benigno y se puede omitir. Desaparecerá en las versiones posteriores de Podman.

13. Este comando compila una imagen de contenedor MySQL específica basándose en la carpeta de contexto **/home/student/D0180/labs/multicontainer-design/images/mysql** y el archivo **Dockerfile** dentro de la carpeta. Espere hasta que se complete la compilación y, luego, ejecute el siguiente comando para verificar que la imagen se haya compilado correctamente:

```
[student@workstation mysql]$ sudo podman images
REPOSITORY                                TAG      IMAGE ID      CREATED       SIZE
localhost/do180/mysql-57-rhel7             latest   8dc111531fce  21 seconds ago  444MB
registry.[...]/rhel7/mysql-57-rhel7        latest   c07bf25398f4  4 weeks ago    444MB
```

► 2. Compile la imagen principal de Node.js con el Dockerfile proporcionado.

- 2.1. Revise el Dockerfile.

Con su editor de preferencia, abra y examine el Dockerfile completo, ubicado en **/home/student/D0180/labs/multicontainer-design/images/nodejs/Dockerfile**.

Observe las siguientes instrucciones definidas en el Dockerfile:

- Se definen dos variables de entorno, **NODEJS_VERSION** y **HOME**, con el comando **ENV**.
- Los paquetes necesarios para Node.js se instalan con **yum** mediante el comando **RUN**.
- Se crean un usuario y un grupo nuevos para ejecutar la aplicación Node.js junto con el directorio **app-root** mediante el comando **RUN**.
- El script **enable-rh-nodejs8.sh**, para ejecutarse automáticamente en el inicio de sesión, se agrega a **/etc/profile.d/** con el comando **ADD**.
- El comando **USER** se usa para cambiar a la cuenta **appuser** creada recientemente.
- El comando **WORKDIR** se usa para cambiar al directorio **\$HOME** para la ejecución de la aplicación.

- 2.2. Compile la imagen principal.

Para compilar la imagen base, ejecute el comando **podman build**.

```
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-design/images/nodejs
[student@workstation nodejs]$ sudo podman build -t do180/nodejs \
> --layers=false .
STEP 1: FROM ubi7/ubi:7.7
Getting image source signatures
...output omitted...
--> Finished Dependency Resolution

Dependencies Resolved
```

```
=====
 Package          Arch    Version      Repository      Size
=====
Installing:
  rh-nodejs8        x86_64  3.0-5.el7   ubi-server-rhscl-7-rpms  7.3 k
...output omitted...
Writing manifest to image destination
Storing signatures
--> 5e61...30de
```

- 2.3. Espere hasta que se complete la compilación y, luego, ejecute el siguiente comando para verificar que la imagen se haya compilado correctamente. Varias columnas de la columna **podman images** no son relevantes, por lo que puede usar la opción **--format** para formatear la salida.

```
[student@workstation nodejs]$ sudo podman images \
> --format "table {{.ID}} {{.Repository}} {{.Tag}}"
IMAGE ID      REPOSITORY           TAG
78c2e6304f23  localhost/do180/mysql-57-rhel7      latest
5e610ea8d94a  localhost/do180/nodejs      latest
c07bf25398f4  registry.access.redhat.com/rhscl/mysql-57-rhel7  latest
22ba71124135  registry.access.redhat.com/ubi7/ubi       7.7
```

- ▶ 3. Compile la imagen secundaria de la aplicación To Do List (Lista de tareas) con el Dockerfile proporcionado.
- 3.1. Revise el Dockerfile.
Con su editor de preferencia, abra y examine el Dockerfile completo, ubicado en **/home/student/D0180/labs/multicontainer-design/deploy/nodejs/Dockerfile**.
 - 3.2. Explore las variables de entorno.
Inspeccione las variables de entorno que permiten al contenedor de la API REST Node.js comunicarse con el contenedor de MySQL.
 - 3.2.1. Observe el archivo **/home/student/D0180/labs/multicontainer-design/deploy/nodejs/nodejs-source/models/db.js**, que contiene la configuración de la base de datos proporcionada a continuación:

```
module.exports.params = {
  dbname: process.env.MYSQL_DATABASE,
  username: process.env.MYSQL_USER,
  password: process.env.MYSQL_PASSWORD,
  params: {
    host: "10.88.100.101",
    port: "3306",
    dialect: 'mysql'
  }
};
```

3.2.2. Observe las variables del entorno que usa la API REST. Esas variables se exponen en el contenedor mediante las opciones **-e** con el comando **podman run** en este ejercicio guiado. Debajo se describen esas variables del entorno.

MYSQL_DATABASE

El nombre de la base de datos MySQL en el contenedor **mysql**.

MYSQL_USER

El nombre del usuario de la base de datos que el contenedor **todoapi** usa para ejecutar los comandos de MySQL.

MYSQL_PASSWORD

La contraseña del usuario de la base de datos que el contenedor **todoapi** usa para la autenticación del contenedor **mysql**.



nota

Los detalles del host y puerto del contenedor de MySQL se incorporan a la aplicación de API REST. El host, como se muestra arriba del archivo **db.js**, es la dirección IP del contenedor **mysql**.

3.3. Compile la imagen secundaria.

Examine el script **/home/student/D0180/labs/multicontainer-design/deploy/nodejs/build.sh** para ver el modo en que se creó la imagen. Ejecute los siguientes comandos para compilar la imagen secundaria.

```
[student@workstation nodejs]$ cd ~/D0180/labs/multicontainer-design/deploy/nodejs  
[student@workstation nodejs]$ ./build.sh  
STEP 1: FROM do180/nodejs  
STEP 2: MAINTAINER username <username@example.com>  
STEP 3: COPY run.sh build ${HOME}/  
...output omitted...  
Writing manifest to image destination  
Storing signatures  
--> 2b12...6463
```



nota

El script **build.sh** reduce las restricciones para el acceso de escritura al directorio de compilación, lo que permite a los usuarios que no son root instalar dependencias.

3.4. Espere hasta que se complete la compilación y, luego, ejecute el siguiente comando para verificar que la imagen se haya compilado correctamente:

```
[student@workstation nodejs]$ sudo podman images \  
> --format "table {{.ID}} {{.Repository}} {{.Tag}}"  
IMAGE ID      REPOSITORY          TAG  
2b127523c28e  localhost/do180/todonodejs    latest  
6bf46c84a3c5   localhost/do180/nodejs        latest  
fa0084527d05   localhost/do180/mysql-57-rhel7  latest  
c07bf25398f4   registry.access.redhat.com/rhscl/mysql-57-rhel7  latest  
22ba71124135   registry.access.redhat.com/ubi7/ubi       7.7
```

- 4. Modifique el script existente para crear contenedores con la IP adecuada, como se definió en el paso anterior. En este script, el orden de los comandos se proporciona de manera que inicie el contenedor **mysql** y, luego, inicie el contenedor **todoapi** antes de conectarlo con el contenedor **mysql**. Después de invocar cada contenedor, hay un tiempo de espera de 9 segundos, para que cada contenedor tenga tiempo de iniciarse.
- 4.1. Edite el archivo **run.sh** ubicado en **/home/student/D0180/labs/multicontainer-design/deploy/nodejs/networked** para insertar el comando **podman run** en la línea adecuada con el fin de invocar el contenedor **mysql**. En la siguiente pantalla se muestra el comando **podman** exacto para insertar en el archivo.

```
sudo podman run -d --name mysql -e MYSQL_DATABASE=items -e MYSQL_USER=user1 \
-e MYSQL_PASSWORD=mypa55 -e MYSQL_ROOT_PASSWORD=r00tpa55 \
-v $PWD/work/data:/var/lib/mysql/data \
-v $PWD/work/init:/var/lib/mysql/init -p 30306:3306 \
--ip 10.88.100.101 do180/mysql-57-rhel7
```

En el comando anterior, **MYSQL_DATABASE**, **MYSQL_USER** y **MYSQL_PASSWORD** se completan con las credenciales para acceder a la base de datos MySQL. Estas variables de entorno son requeridas para que se ejecute el contenedor **mysql**. También, las carpetas locales **\$PWD/work/data** y **\$PWD/work/init** montadas como volúmenes en el sistema de archivos del contenedor.

Observe la IP asignada al contenedor. Esta IP debe ser la misma que la proporcionada en el archivo **/home/student/D0180/labs/multicontainer-design/deploy/nodejs/nodejs-source/models/db.js**.

- 4.2. En el mismo archivo **run.sh**, inserte otro comando **podman run** en la línea adecuada para ejecutar el contenedor **todoapi**. En la siguiente pantalla se muestra el comando **docker** para insertar en el archivo.

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



nota

Después de cada comando **podman run** insertado en el script **run.sh**, asegúrese de que también haya un comando **sleep 9**. Si necesita repetir este paso, debe eliminar el directorio **work** y su contenido antes de volver a ejecutar el script **run.sh**.

- 4.3. Verifique que su script **run.sh** coincida con el script de la solución ubicado en **/home/student/D0180/solutions/multicontainer-design/deploy/nodejs/networked/run.sh**.
- 4.4. Guardar el archivo y salir del editor.
- 5. Ejecute los contenedores.
- 5.1. Use el siguiente comando para ejecutar el script que actualizó a fin de iniciar los contenedores **mysql** y **todoapi**.

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

**nota**

Es posible que la IP seleccionada para el contenedor (10.88.100.101) ya esté reservada para otro contenedor, incluso si ese contenedor ya se ha eliminado. En este caso, elimine la imagen **todonodejs** y el contenedor antes de crear uno actualizado con el comando **sudo podman rmi -f todonodejs**. A continuación, borre el contenedor MySQL con el comando **sudo podman rm -f mysql**. Despues, vuelva al paso 3 y actualice **db.js** y **run.sh** con otra IP disponible.

- 5.2. Verifique que los contenedores se iniciaron correctamente.

```
[student@workstation networked]$ sudo podman ps \
> --format="table {{.ID}} {{.Names}} {{.Image}} {{.Status}}"
CONTAINER ID NAMES IMAGE STATUS
c74b4709e3ae todoapi localhost/do180/todonodejs:latest Up 3 minutes ago
3bc19f74254c mysql localhost/do180/mysql-57-rhel7:latest Up 3 minutes ago
```

- 6. Examine las variables de entorno del contenedor de API.

Ejecute el siguiente comando para explorar las variables de entorno que se encuentran en el contenedor de API.

```
[student@workstation networked]$ sudo podman exec -it todoapi env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=2c61a089105d
TERM=xterm
MYSQL_DATABASE=items
MYSQL_USER=user1
MYSQL_PASSWORD=mypa55
container=oci
NODEJS_VERSION=8.0
HOME=/opt/app-root/src
```

- 7. Pruebe la aplicación.

- 7.1. Ejecute un comando **curl** para probar la API REST para la aplicación To Do List (Lista de tareas).

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

La opción **-w "\n"** con el comando **curl** permite que aparezca el prompt de shell en la siguiente línea en lugar de combinarlo con la salida en la misma línea.

- 7.2. Abra Firefox en **workstation** y vaya a <http://127.0.0.1:30080/todo/>. Debería ver la aplicación To Do List (Lista de tareas).



nota

Asegúrese de agregar la barra final (/).

Finalizar

En **workstation**, ejecute el script **lab multicontainer-design finish** para terminar este ejercicio.

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

Esto concluye el ejercicio guiado.

Implementación de una aplicación con varios contenedores en OpenShift

Objetivos

Tras finalizar esta sección, los estudiantes deberán ser capaces de implementar una aplicación con varios contenedores en OpenShift mediante el uso de una plantilla.

Examen del esqueleto de una plantilla

La implementación de una aplicación en OpenShift Container Platform a menudo requiere la creación de varios recursos relacionados dentro de un proyecto. Por ejemplo, una aplicación web puede requerir un recurso **BuildConfig**, **DeploymentConfig**, **Service** y **Route** para ejecutarse en un proyecto OpenShift. A menudo, los atributos de estos recursos tienen el mismo valor, como el atributo de nombre de un recurso.

Las *plantillas* de OpenShift proporcionan una manera de simplificar la creación de recursos que requiere una aplicación. Una plantilla define un conjunto de recursos relacionados que se crearán juntos, así como un conjunto de parámetros de una aplicación. Por lo general, los atributos de los recursos de plantilla se definen en términos de los parámetros de la plantilla, como el atributo de nombre de un recurso.

Por ejemplo, una aplicación puede consistir en una aplicación web de front-end y un servidor de base de datos. Cada una consiste en un recurso de servicio y un recurso de configuración de implementación. Comparten un conjunto de credenciales (parámetros) para que front-end realice la autenticación para back-end. La plantilla se puede procesar mediante la especificación de parámetros o permitiendo que se generen automáticamente (por ejemplo, para una contraseña de base de datos única) para crear una instancia de la lista de recursos en la plantilla como aplicación coordinada.

El instalador de OpenShift crea varias plantillas de forma predeterminada en el espacio de nombres **openshift**. Ejecute el comando **oc get templates** con la opción **-n openshift** para enumerar estas plantillas preinstaladas:

```
[student@workstation ~]$ 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-pgsql-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, ...
```

A continuación, se muestra la definición de una plantilla YAML:

```
[student@workstation ~]$ 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: DeploymentConfig
  metadata:
    annotations: ...annotations omitted...
    name: ${DATABASE_SERVICE_NAME}
  spec: ...spec 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 Define una lista de etiquetas arbitrarias para asociar con esta plantilla. Ingrese cualquiera de estas etiquetas en la interfaz de usuario para encontrar esta plantilla.
- ➋ Define el nombre de la plantilla.
- ➌ En la sección **objetos**, se define la lista de recursos de OpenShift para esta plantilla. Esta plantilla crea cuatro recursos: **Secret**, **Service**, **PersistentVolumeClaim** y **DeploymentConfig**.
- ➍ Los cuatro objetos de recursos tienen sus nombres establecidos en el valor del parámetro **DATABASE_SERVICE_NAME**.
- ➎➏ La sección **parámetros** contiene una lista de nueve parámetros. Los recursos de plantilla a menudo definen sus atributos a través de los valores de estos parámetros, como se demuestra con el parámetro **DATABASE_SERVICE_NAME**.
- ➐ Si no especifica un valor para el parámetro **MYSQL_PASSWORD** cuando crea una aplicación con esta plantilla, OpenShift genera una contraseña que coincide con esta expresión regular.

Puede publicar una nueva plantilla en el clúster de OpenShift, de modo que otros desarrolladores puedan compilar una aplicación desde la plantilla.

Supongamos que tiene una aplicación de lista de tareas denominada **todo** que requiere un objeto **DeploymentConfig**, **Service** y **Route** para la implementación. Crea un archivo de definición de plantilla YAML que define los atributos para estos recursos de OpenShift, junto con definiciones para cualquier parámetro requerido. Suponiendo que la plantilla se define en el archivo **todo-template.yaml**, use el comando **oc create** para publicar la plantilla de la aplicación:

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

De forma predeterminada, la plantilla se crea en el proyecto actual, a menos que especifique uno diferente con la opción **-n**, como se muestra en el siguiente ejemplo:

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



Importante

Cualquier plantilla creada bajo el espacio de nombres **openshift** (proyecto OpenShift) está disponible en la consola web bajo el cuadro de diálogo al que se accede a través del ítem de menú **Catalog (Catálogo) → Developer Catalog (Catálogo de desarrollador)**. Es más, se podrá acceder a cualquier plantilla que se cree bajo el proyecto actual desde el mismo proyecto.

Parámetros

Las plantillas definen un conjunto de *parámetros*, a los cuales se asignan valores. Los recursos de OpenShift definidos en la plantilla pueden obtener sus valores de configuración mediante la referencia a los *parámetros nombrados*. Los parámetros en una plantilla pueden tener valores

predeterminados, pero son opcionales. Todos los valores predeterminados pueden reemplazarse al procesar la plantilla.

El valor de cada parámetro puede establecerse de forma explícita mediante el uso del comando **oc process**, o puede ser generado por OpenShift de acuerdo con la configuración del parámetro.

Hay dos maneras de mostrar los parámetros disponibles desde una plantilla. La primera es mediante el comando **oc describe**:

```
$ 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 (5.7, or latest).
Required: true
Value: 5.7

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}
DeploymentConfig    ${DATABASE_SERVICE_NAME}
```

La segunda manera es mediante **oc process** con la opción **--parameters**:

```
$ 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...

```

Procesamiento de una plantilla con la CLI

Cuando procesa una plantilla, genera una lista de recursos para crear una nueva aplicación. Para procesar una plantilla, use el comando **oc process**:

```
$ oc process -f <filename>
```

El comando anterior procesa un archivo de plantilla, con formato JSON o YAML, y devuelve la lista de recursos en la salida estándar. El formato de la lista de recursos de salida es JSON. Para generar la lista de recursos en formato YAML, use **-o yaml** con el comando **oc process**:

```
$ oc process -o yaml -f <filename>
```

Otra opción es procesar una plantilla desde el proyecto actual o el proyecto **openshift**:

```
$ oc process <uploaded-template-name>
```



nota

El comando **oc process** muestra una lista de recursos en la salida estándar. Esta salida puede redirigirse a un archivo:

```
$ oc process -o yaml -f filename > myapp.yaml
```

Las plantillas a menudo generan recursos con atributos configurables que se basan en los parámetros de la plantilla. Para reemplazar un parámetro, use la opción **-p** seguida de un par **<name>=<value>**.

```
$ 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
```

Para crear la aplicación, use el archivo de definición de recursos YAML generado:

```
$ oc create -f mysqlProcessed.yaml
```

Como alternativa, es posible procesar la plantilla y crear la aplicación sin guardar un archivo de definición de recursos mediante el uso de una tubería UNIX:

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

Para usar una plantilla en el proyecto **openshift** para crear una aplicación en su proyecto, primero exporte la plantilla:

```
$ oc get template mysql-persistent -o yaml \
> -n openshift > mysql-persistent-template.yaml
```

Luego, identifique los valores adecuados para los parámetros de la plantilla y procese la plantilla:

```
$ 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 -
```

También puede usar dos barras oblicuas (//) para proporcionar el espacio de nombres como parte del nombre de la plantilla:

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

Como alternativa, es posible crear una aplicación con el comando **oc new-app**, pasando el nombre de la plantilla como el argumento de la opción **--template**:

```
$ oc new-app --template=mysql-persistent \
> -p MYSQL_USER=dev -p MYSQL_PASSWORD=$P4SSD -p MYSQL_DATABASE=bank \
> -p VOLUME_CAPACITY=10Gi \
> --as-deployment-config
```

Configuración del almacenamiento persistente para aplicaciones OpenShift

OpenShift Container Platform administra el almacenamiento persistente como un conjunto de recursos agrupados en todo el clúster. Para agregar un recurso de almacenamiento al clúster, un administrador de OpenShift crea un objeto **PersistentVolume** que define los metadatos necesarios para el recurso de almacenamiento. Los metadatos describen cómo el clúster accede

al almacenamiento, así como a otros atributos de almacenamiento, como la capacidad o el rendimiento.

Para enumerar los objetos **PersistentVolume** en un clúster, use el comando **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...
```

Para ver la definición de YAML para un determinado **PersistentVolume**, use el comando **oc get** con la opción **-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
```

Para agregar más objetos **PersistentVolume** a un clúster, use el comando **oc create**:

```
[admin@host ~]$ oc create -f pv1001.yaml
```



nota

El archivo **pv1001.yaml** anterior debe contener una definición de volumen persistente, con una estructura similar a la salida del comando **oc get pv pv-name -o yaml**.

Solicitud de volúmenes persistentes

Cuando una aplicación requiere almacenamiento, crea un objeto **PersistentVolumeClaim** (PVC) para solicitar un recurso de almacenamiento dedicado del pool (grupo) de clústeres. El siguiente contenido de un archivo denominado **pvc.yaml** es una definición de ejemplo para un PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myapp
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

El PVC define los requisitos de almacenamiento para la aplicación, como la capacidad o el rendimiento. Para crear el PVC, use el comando **oc create**:

```
[admin@host ~]$ oc create -f pvc.yaml
```

Después de crear un PVC, OpenShift intenta encontrar un recurso **PersistentVolume** disponible que cumpla con los requisitos del PVC. Si OpenShift encuentra una coincidencia, vincula el objeto PersistentVolume al objeto PersistentVolumeClaim. Para enumerar los PVC de un proyecto, use el comando **oc get pvc**:

```
[admin@host ~]$ oc get pvc
NAME      STATUS    VOLUME   CAPACITY  ACCESS MODES  STORAGECLASS  AGE
myapp    Bound     pv0001   1Gi       RWO          6s
```

La salida indica si un volumen persistente está vinculado al PVC, junto con los atributos del PVC (como la capacidad).

Para usar el volumen persistente en un pod de aplicación, defina un montaje de volumen para un contenedor que haga referencia al objeto **PersistentVolumeClaim**. La definición del pod de aplicación a continuación hace referencia a un objeto **PersistentVolumeClaim** para definir un montaje de volumen para la aplicación:

```
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 En esta sección, se declara que el volumen **pvol** se monta en **/var/www/html** en el sistema de archivos del contenedor.
- ② En esta sección, se define el volumen **pvol**.
- ③ El volumen **pvol** hace referencia al PVC **myapp**. Si OpenShift asocia un volumen persistente disponible al PVC **myapp**, el volumen **pvol** se refiere a este volumen asociado.

Configuración del almacenamiento persistente con plantillas

Las plantillas se usan a menudo para simplificar la creación de aplicaciones que requieren almacenamiento persistente. Muchas de estas plantillas tienen un sufijo **-persistent**:

```
[student@workstation ~]$ oc get templates -n openshift | grep persistent
cakephp-mysql-persistent   An example CakePHP application with a MySQL data...
dancer-mysql-persistent    An example Dancer application with a MySQL datab...
django-psql-persistent     An example Django application with a PostgreSQL ...
dotnet-psql-persistent     An example .NET Core application with a Postgres...
jenkins-persistent          Jenkins service, with persistent storage....
mariadb-persistent          MariaDB database service, with persistent storag...
mongodb-persistent          MongoDB database service, with persistent storag...
mysql-persistent             MySQL database service, with persistent storage...
nodejs-mongo-persistent     An example Node.js application with a MongoDB da...
postgresql-persistent       PostgreSQL database service, with persistent sto...
rails-psql-persistent       An example Rails application with a PostgreSQL d...
redis-persistent              Redis in-memory data structure store, with persi...
```

En la siguiente plantilla de ejemplo, se define un objeto **PersistentVolumeClaim** junto con un objeto **DeploymentConfig**:

```

apiVersion: template.openshift.io/v1
kind: Template
labels:
  template: myapp-persistent-template
metadata:
  name: myapp-persistent
  namespace: openshift
objects:
- apiVersion: v1
  kind: PersistentVolumeClaim ①
  metadata:
    name: ${APP_NAME}
  spec:
    accessModes:
      - ReadWriteOnce
    resources:
      requests:
        storage: ${VOLUME_CAPACITY}

```

```

- apiVersion: v1
kind: DeploymentConfig ②
metadata:
  name: ${APP_NAME}
spec:
  replicas: 1
  selector:
    name: ${APP_NAME}
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        name: ${APP_NAME}
    spec:
      containers:
        - image: 'openshift/myapp'
          name: myapp
          volumeMounts:
            - mountPath: /var/lib/myapp/data
              name: ${APP_NAME}-data ③
      volumes:
        - name: ${APP_NAME}-data ④
          persistentVolumeClaim:
            claimName: ${APP_NAME}
  parameters:
    - description: The name for the myapp application.
      displayName: Application Name
      name: APP_NAME ⑤
      required: true
      value: myapp
    - description: Volume space available for data, e.g. 512Mi, 2Gi.
      displayName: Volume Capacity
      name: VOLUME_CAPACITY ⑥
      required: true
      value: 1Gi

```

① ② En la plantilla de ejemplo, se definen los objetos **PersistentVolumeClaim** y **DeploymentConfig**. Ambos objetos tienen nombres que coinciden con el valor del parámetro **APP_NAME**. La reclamación de volumen persistente define una capacidad correspondiente al valor del parámetro **VOLUME_CAPACITY**.

③ ④ El objeto **DeploymentConfig** define un montaje de volumen que hace referencia a **PersistentVolumeClaim** creado por la plantilla.

⑤ ⑥ La plantilla define dos parámetros: **APP_NAME** y **VOLUME_CAPACITY**. La plantilla usa estos parámetros para especificar el valor de los atributos para los objetos **PersistentVolumeClaim** y **DeploymentConfig**.

Con esta plantilla, solo necesita especificar los parámetros **APP_NAME** y **VOLUME_CAPACITY** para desplegar la aplicación **myapp** con almacenamiento persistente:

```
[student@workstation ~]$ oc create myapp-template.yaml
template.template.openshift.io/myapp-persistent created
[student@workstation ~]$ oc process myapp-persistent \
> -p APP_NAME=myapp-dev -p VOLUME_CAPACITY=1Gi \
> | oc create -f -
deploymentconfig/myapp created
persistentvolumeclaim/myapp created
```



Referencias

Puede encontrar información para desarrolladores acerca de las plantillas en la sección *Using Templates* (Uso de plantillas) de la documentación de OpenShift Container Platform:

Guía para desarrolladores

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/images/index#using-templates

► Ejercicio Guiado

Creación de una aplicación con una plantilla

En este ejercicio, implementará la aplicación To Do List (Lista de tareas) en OpenShift Container Platform con una plantilla para definir los recursos que su aplicación necesita para ejecutarse.

Resultados

Debería poder compilar e implementar una aplicación en OpenShift Container Platform con una plantilla JSON provista.

Andes De Comenzar

Debe tener el código fuente de la aplicación To Do List (Lista de tareas) y los archivos del trabajo de laboratorio en **workstation**. Para descargar los archivos del trabajo de laboratorio y verificar el estado del clúster de OpenShift, ejecute el siguiente comando en una nueva ventana de terminal.

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

- 1. Use **Dockerfile** en el subdirectorio **images/mysql** para compilar el contenedor de la base de datos. Publique la imagen de contenedor en `quay.io` con la etiqueta **do180-mysql-57-rhel7**.

- 1.1. Compile la imagen de la base de datos MySQL.

```
[student@workstation ~]$ cd ~/DO180/labs/multicontainer-openshift/images/mysql
[student@workstation mysql]$ sudo podman build -t do180-mysql-57-rhel7 .
STEP 1: FROM rhscl/mysql-57-rhel7
Getting image source signatures
Copying blob sha256:e373541...output omitted...
69.66 MB / 69.66 MB [=====] 6s
Copying blob sha256:c5d2e94...output omitted...
1.20 KB / 1.20 KB [=====] 0s
Copying blob sha256:b3949ae...output omitted...
62.03 MB / 62.03 MB [=====] 5s
Writing manifest to image destination
Storing signatures
STEP 2: ADD root /
ERROR[0001] HOSTNAME is not supported for OCI image format ...output omitted...
--> b628...a079
STEP 3: COMMIT do180-mysql-57-rhel7
```

**nota**

El mensaje de error **ERRO** es un problema conocido con una versión anterior de Podman. Puede ignorar este mensaje.

- 1.2. Para hacer que la imagen esté disponible para OpenShift, etiquétala y envíela a quay.io. Para ello, en la ventana de terminal, ejecute los siguientes comandos.

```
[student@workstation mysql]$ source /usr/local/etc/ocp4.config
[student@workstation mysql]$ sudo podman login quay.io -u ${RHT_OCP4_QUAY_USER}
Password: your_quay_password
Login Succeeded!
[student@workstation mysql]$ sudo podman tag \
> do180-mysql-57-rhel7 quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-57-rhel7
[student@workstation mysql]$ sudo podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-57-rhel7
Getting image source signatures
Copying blob ssha256:04dbae...b21619
205.17 MB / 205.17 MB [=====] 1m19s
...output omitted...
Writing manifest to image destination
Storing signatures
```

**nota**

El valor **config sha256**: en la salida anterior puede diferir de su salida.

- ▶ 2. Compile la imagen base de la aplicación To Do List (Lista de tareas) con el Dockerfile Node.js ubicado en el subdirectorio del ejercicio **images/nodejs**. Etiquete la imagen como **do180-nodejs**. No publique esta imagen en el registro.

```
[student@workstation mysql]$ cd ~/DO180/labs/multicontainer-openshift
[student@workstation multicontainer-openshift]$ cd images/nodejs
[student@workstation nodejs]$ sudo podman build -t do180-nodejs .
STEP 1: ubi7/ubi:7.7
Getting image source signatures
Copying blob sha256:5d92fc...1ce84e
...output omitted...
Storing signatures
STEP 2: MAINTAINER username <username@example.com>
...output omitted...
STEP 14: CMD ["echo", "You must create your own container from this one."]
--> 97f0eb3...output omitted...
STEP 15: COMMIT do180-nodejs
```

- ▶ 3. Use el script **build.sh** en el subdirectorio **deploy/nodejs** para compilar la aplicación To Do List (Lista de tareas). Publique la imagen de la aplicación en quay.io con una etiqueta de imagen de **do180-todonodejs**.

- 3.1. Vaya al directorio **~/DO180/labs/multicontainer-openshift/deploy/nodejs** y ejecute el comando **build.sh** para compilar la imagen secundaria.

```
[student@workstation nodejs]$ cd ~/DO180/labs/multicontainer Openshift
[student@workstation multicontainer Openshift]$ cd deploy/nodejs
[student@workstation nodejs]$ ./build.sh
Preparing build folder
STEP 1: FROM do180-nodejs
STEP 2: MAINTAINER username <username@example.com>
...output omitted...
STEP 7: CMD ["scl","enable","rh-nodejs8","./run.sh"]
--> f627d64...output omitted...
STEP 8: COMMIT containers-storage:...
```

3.2. Envíe la imagen a quay.io.

Para hacer que la imagen esté disponible para que OpenShift la use en la plantilla, etiquétela y envíela al registro privado. Para ello, en la ventana de terminal, ejecute los siguientes comandos.

```
[student@workstation nodejs]$ sudo podman tag do180/todonodejs \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-todonodejs
[student@workstation nodejs]$ sudo podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-todonodejs
Getting image source signatures
Copying blob sha256:24a5c62...output omitted...
...output omitted...
Copying blob sha256:5f70bf1...output omitted...
1024 B / 1024 B [=====] 0s
Copying config sha256:c43306d...output omitted...
7.26 KB / 7.26 KB [=====] 0s
Writing manifest to image destination
Copying config sha256:c43306d...output omitted...
0 B / 7.26 KB [-----] 0s
Writing manifest to image destination
Storing signatures
```



Advertencia

Asegúrese de que los dos repositorios sean públicos en quay.io, de modo que OpenShift pueda obtener las imágenes de este. Consulte la sección de **Repositories Visibility** (Visibilidad de repositorios) del Apéndice C para conocer los detalles acerca de cómo cambiar la visibilidad del repositorio.

► 4. Cree la aplicación To Do List (Lista de tareas) desde la plantilla JSON provista.

4.1. Inicie sesión en OpenShift Container Platform.

```
[student@workstation nodejs]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.

...output omitted...

Using project "default".
```

Si el comando **oc login** muestra un prompt acerca del uso de conexiones no seguras, responda **y** (sí).

- 4.2. Cree una *plantilla* de proyecto nueva en OpenShift a fin de usarla para este ejercicio. Ejecute el siguiente comando para crear la **plantilla** del proyecto.

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

- 4.3. Revise la plantilla.

Con su editor de preferencia, abra y examine la plantilla ubicada en **/home/student/D0180/labs/multicontainer-openshift/todo-template.json**. Observe los siguientes recursos definidos en la plantilla y revise sus configuraciones.

- La definición del pod **todoapi** establece la aplicación Node.js.
- La definición del pod **mysql** establece la base de datos MySQL.
- El servicio **todoapi** proporciona conectividad al pod de la aplicación Node.js.
- El servicio **mysql** proporciona conectividad al pod de la base de datos MySQL.
- La definición de reclamación de volumen persistente **dbinit** establece el volumen **/var/lib/mysql/init** de MySQL.
- La definición de reclamación de volumen persistente **dbcclaim** establece el volumen **/var/lib/mysql/data** de MySQL.

- 4.4. Procese la plantilla y cree los recursos de la aplicación.

Use el comando **oc process** para procesar el archivo de plantillas. Esta plantilla requiere el espacio de nombres de Quay.io para recuperar las imágenes de contenedor, como el parámetro **RHT_OCP4_QUAY_USER**. Use el comando **pipe** para enviar el resultado al comando **oc create**.

En la ventana de terminal, ejecute el siguiente comando:

```
[student@workstation nodejs]$ cd /home/student/D0180/labs/multicontainer-openshift
[student@workstation multicontainer-openshift]$ oc process \
> -f todo-template.json -p RHT_OCP4_QUAY_USER=${RHT_OCP4_QUAY_USER} \
> | oc create -f -
pod/mysql created
pod/todoapi created
service/todoapi created
service/mysql created
persistentvolumeclaim/dbinit created
persistentvolumeclaim/dbclaim created
```

- 4.5. Revise la implementación.

Revise el estado de la implementación mediante el comando **oc get pods** con la opción **-w** para continuar monitoreando el estado del pod. Espere hasta que ambos contenedores estén ejecutándose. El inicio de ambos pods puede llevar tiempo.

```
[student@workstation multicontainer-openshift]$ oc get pods -w
NAME      READY   STATUS          RESTARTS   AGE
mysql     0/1    ContainerCreating   0          9s
todoapi   1/1    Running         0          9s
mysql     1/1    Running         0          2m
```

Presione **Ctrl+C** para salir del comando.

► 5. Exponga el servicio.

Para permitir el acceso a la aplicación To Do List (Lista de tareas) a través del enrutador de OpenShift y permitir que esté disponible como FQDN público, use el comando **oc expose** para exponer el servicio **todoapi**.

En la ventana de terminal, ejecute el siguiente comando.

```
[student@workstation multicontainer-openshift]$ oc expose service todoapi
route.route.openshift.io/todoapi exposed
```

► 6. Pruebe la aplicación.

- 6.1. Busque el FQDN de la aplicación ejecutando el comando **oc status** y observe el FQDN para la aplicación.

En la ventana de terminal, ejecute el siguiente comando.

```
[student@workstation multicontainer-openshift]$ oc status
In project template on server ...

svc/mysql - 172.30.77.172:3306
pod/mysql runs quay.io/your_quay_username/do180-mysql-57-rhel7

http://todoapi-your_quay_username-template.your_ocp4_wildcard_domain to pod port
30080 (svc/todoapi)
pod/todoapi runs quay.io/your_quay_username/do180-todonodejs

2 infos identified, use 'oc status --suggest' to see details.
```

- 6.2. Use **curl** para probar la API REST para la aplicación To Do List (Lista de tareas).

```
[student@workstation multicontainer-openshift]$ curl -w "\n" \
http://your_quay_username-todoapi-template.your_ocp4_wildcard_domain/todo/api/
items/1
{"id":1,"description":"Pick up newspaper","done":false}
```

La opción **-w "\n"** con el comando **curl** permite que aparezca el prompt de shell en la siguiente línea en lugar de combinarlo con la salida en la misma línea.

- 6.3. Abra Firefox en **workstation** y apunte su navegador a **http://todoapi-your_quay_username-template.your_ocp4_wildcard_domain/todo/**; debería ver la aplicación To Do List (Lista de tareas).

**nota**

Se necesita la barra final de la URL mencionada anteriormente. Si no incluye esto en la URL, puede encontrar problemas con la aplicación.

Id	Description	Done
1	Pick up newspaper	false
2	Buy groceries	true

Figura 7.4: Aplicación To Do List

Finalizar

En **workstation**, ejecute el script **lab multicontainer-openshift finish** para terminar este trabajo de laboratorio.

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

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Implementación de aplicaciones con múltiples contenedores

Lista de verificación de rendimiento

En este trabajo de laboratorio, implementará una aplicación PHP con una base de datos MySQL mediante una plantilla de OpenShift para definir los recursos necesarios para la aplicación.

Resultados

Deberá ser capaz de crear una aplicación de OpenShift compuesta por varios contenedores y acceder a esta a través de un navegador web.

Andes De Comenzar

Abra un terminal **workstation** como el usuario **student** y ejecute los siguientes comandos:

```
[student@workstation ~]$ lab multicontainer-review start
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-review
```

1. Inicie sesión en el clúster de OpenShift y cree un nuevo proyecto para este ejercicio.
2. Compile la imagen de contenedor de bases de datos ubicada en el directorio **images/mysql** y publíquela en el repositorio Quay.io.
3. Compile la imagen de contenedor de PHP ubicada en **images/quote-php** y publíquela en el repositorio Quay.io.



Advertencia

Asegúrese de que los dos repositorios sean públicos en `quay.io`, de modo que OpenShift pueda obtener las imágenes de este. Consulte la sección de **Repositories Visibility** (Visibilidad de repositorios) del Apéndice C para conocer los detalles acerca de cómo cambiar la visibilidad del repositorio.

4. Diríjase al directorio `/home/student/D0180/labs/multicontainer-review/` y revise el archivo de plantillas `quote-php-template.json` provisto.
Observe las definiciones y la configuración de los pods, los servicios y las recuperaciones de volúmenes persistentes definidos en la plantilla.
5. Cargue la plantilla de la aplicación PHP para que pueda usarla cualquier desarrollador con acceso a su proyecto.
6. Procese la plantilla cargada y cree los recursos de la aplicación.
7. Exponga el servicio.
8. Pruebe la aplicación y verifique que emita un mensaje inspirador.

Evaluación

Evalúe su trabajo ejecutando el comando **lab multicontainer-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

Para completar este trabajo de laboratorio, ejecute el comando **lab multicontainer-review finish** en **workstation**.

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

Esto concluye el trabajo de laboratorio.

► Solución

Implementación de aplicaciones con múltiples contenedores

Lista de verificación de rendimiento

En este trabajo de laboratorio, implementará una aplicación PHP con una base de datos MySQL mediante una plantilla de OpenShift para definir los recursos necesarios para la aplicación.

Resultados

Deberá ser capaz de crear una aplicación de OpenShift compuesta por varios contenedores y acceder a esta a través de un navegador web.

Andes De Comenzar

Abra un terminal **workstation** como el usuario **student** y ejecute los siguientes comandos:

```
[student@workstation ~]$ lab multicontainer-review start
[student@workstation ~]$ cd ~/DO180/labs/multicontainer-review
```

1. Inicie sesión en el clúster de OpenShift y cree un nuevo proyecto para este ejercicio.

- 1.1. En **workstation**, inicie sesión con el usuario provisto en el primer ejercicio.

```
[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.

...output omitted...
```

Using project "default".

Si el comando **oc login** muestra un prompt acerca del uso de conexiones no seguras, responda **y** (sí).

- 1.2. Cree un nuevo proyecto en OpenShift denominado **deploy** y precedido por su nombre de usuario de OpenShift:

```
[student@workstation multicontainer-review]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-deploy
Now using project ...output omitted...
```

2. Compile la imagen de contenedor de bases de datos ubicada en el directorio **images/mysql** y publíquela en el repositorio Quay.io.

- 2.1. Compile la imagen de la base de datos MySQL con el Dockerfile provisto en el directorio **images/mysql**.

```
[student@workstation multicontainer-review]$ cd images/mysql
[student@workstation mysql]$ sudo podman build -t do180-mysql-57-rhel7 .
STEP 1: FROM rhscl/mysql-57-rhel7
...output omitted...
STEP 5: COMMIT do180-mysql-57-rhel7
```

- 2.2. Envíe la imagen de MySQL al repositorio de su Quay.io.

Para hacer que la imagen esté disponible para que OpenShift la use en la plantilla, asígnale una etiqueta de **quay.io/\${RHT_OCP4_QUAY_USER}/do180-mysql-57-rhel7** y envíela al registro de quay.io. Para enviar imágenes a quay.io, primero debe iniciar sesión con sus propias credenciales.

```
[student@workstation mysql]$ sudo podman login quay.io -u ${RHT_OCP4_QUAY_USER}
Password: your_quay_password
Login Succeeded!
```

Para etiquetar y enviar la imagen, ejecute los siguientes comandos en la ventana de terminal.

```
[student@workstation mysql]$ sudo podman tag do180-mysql-57-rhel7 \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-57-rhel7
[student@workstation mysql]$ sudo podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-57-rhel7
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

Regrese al directorio anterior.

```
[student@workstation mysql]$ cd ~/DO180/labs/multicontainer-review
```

3. Compile la imagen de contenedor de PHP ubicada en **images/quote-php** y publíquela en el repositorio Quay.io.



Advertencia

Asegúrese de que los dos repositorios sean públicos en quay.io, de modo que OpenShift pueda obtener las imágenes de este. Consulte la sección de **Repositories Visibility** (Visibilidad de repositorios) del Apéndice C para conocer los detalles acerca de cómo cambiar la visibilidad del repositorio.

- 3.1. Compile la imagen PHP con el Dockerfile provisto en el directorio **images/quote-php**.

```
[student@workstation multicontainer-review]$ cd images/quote-php  
[student@workstation quote-php]$ sudo podman build -t do180-quote-php .  
STEP 1: FROM ubi7/ubi:7.7  
...output omitted...  
STEP 15: COMMIT do180-quote-php
```

- 3.2. Etiquete la imagen PHP y envíela a su registro de Quay.io.

Para hacer que la imagen esté disponible para que OpenShift la use en la plantilla, asignele una etiqueta de **quay.io/\${RHT_OCP4_QUAY_USER}/do180-quote-php** y envíela a Quay.io.

```
[student@workstation quote-php]$ sudo podman tag do180-quote-php \  
> quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php  
[student@workstation quote-php]$ sudo podman push \  
> quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php  
Getting image source signatures  
...output omitted...  
Writing manifest to image destination  
Storing signatures
```

4. Diríjase al directorio **/home/student/D0180/labs/multicontainer-review/** y revise el archivo de plantillas **quote-php-template.json** provisto.

Observe las definiciones y la configuración de los pods, los servicios y las recuperaciones de volúmenes persistentes definidos en la plantilla.

```
[student@workstation quote-php]$ cd ~/D0180/labs/multicontainer-review
```

5. Cargue la plantilla de la aplicación PHP para que pueda usarla cualquier desarrollador con acceso a su proyecto.

Use el comando **oc create -f** para cargar el archivo de plantillas para el proyecto.

```
[student@workstation multicontainer-review]$ oc create -f quote-php-template.json  
template.template.openshift.io/quote-php-persistent created
```

6. Procese la plantilla cargada y cree los recursos de la aplicación.

- 6.1. Use el comando **oc process** para procesar el archivo de plantillas. Asegúrese de proporcionar el parámetro **RHT_OCP4_QUAY_USER** con el espacio de nombres de **quay.io** en el que se encuentran las imágenes. Use el comando **pipe** para enviar el resultado al comando **oc create** con el fin de crear una aplicación desde la plantilla.

```
[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. Verifique el estado de la implementación mediante el comando **oc get pods** con la opción **-w** para monitorear el estado de la implementación. Espere hasta que ambos pods estén en ejecución. El inicio de ambos pods puede llevar tiempo.

```
[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
```

Presione **Ctrl+C** para salir del comando.

7. Exponga el servicio.

Para permitir el acceso a la aplicación PHP Quote a través del enrutador de OpenShift y para permitir que esté disponible desde una red externa, use el comando **oc expose** para exponer el servicio **quote-php**.

En la ventana de terminal, ejecute el siguiente comando.

```
[student@workstation multicontainer-review]$ oc expose svc quote-php
route.route.openshift.io/quote-php exposed
```

8. Pruebe la aplicación y verifique que emita un mensaje inspirador.

- 8.1. Use el comando **oc get route** para encontrar el FQDN donde la aplicación está disponible. Observe el FQDN para la aplicación.

En la ventana de terminal, ejecute el siguiente comando.

```
[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. Use el comando **curl** para probar la API REST para la aplicación 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.
```



nota

El texto que se muestra en la salida anterior puede diferir, pero el comando **curl** debe ejecutarse correctamente.

Evaluación

Evalué su trabajo ejecutando el comando **lab multicontainer-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

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

Finalizar

Para completar este trabajo de laboratorio, ejecute el comando **lab multicontainer-review finish** en **workstation**.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Las redes definidas por software permiten la comunicación entre contenedores. Los contenedores deben estar unidos a la misma red definida por software para que puedan comunicarse.
- Las aplicaciones en contenedores no pueden depender de direcciones IP fijas o nombres de host para buscar servicios.
- Podman usa la Interfaz de red de contenedor (CNI) para crear una red definida por software y une todos los contenedores del host a esa red. Kubernetes y OpenShift crean una red definida por software entre todos los contenedores de un pod.
- Dentro del mismo proyecto, Kubernetes inserta un conjunto de variables para cada servicio en todos los pods.
- Las plantillas de Kubernetes automatizan la creación de aplicaciones que consisten en varios recursos. Los parámetros de plantilla permiten usar los mismos valores al crear múltiples recursos.

capítulo 8

Solución de problemas de aplicaciones en contenedores

Meta

Solucione los problemas de una aplicación en contenedores implementada en OpenShift.

Objetivos

- Solucionar problemas de implementación y compilación de aplicaciones en OpenShift.
- Implementar técnicas para solucionar problemas y depurar las aplicaciones en contenedores.

Secciones

- Solución de problemas de implementaciones y compilaciones con S2I (y ejercicio guiado)
- Solución de problemas de aplicaciones en contenedores (y ejercicio guiado)

Trabajo de laboratorio

- Solución de problemas de aplicaciones contenerizadas

Solución de problemas de implementaciones y compilaciones con S2I

Objetivos

Tras finalizar esta sección, usted deberá ser capaz de realizar lo siguiente:

- Solucionar problemas en los pasos de implementación y compilación de aplicaciones en OpenShift.
- Analizar registros de OpenShift para identificar problemas durante el proceso de implementación y compilación.

Introducción al Proceso S2I

El proceso *Source-to-Image* (*S2I*) es una manera simple de crear automáticamente imágenes sobre la base del lenguaje de programación del código fuente de la aplicación en OpenShift. Aunque este proceso a menudo es una manera conveniente de implementar aplicaciones con rapidez, pueden surgir problemas durante el proceso de creación de imágenes *S2I*, ya sea mediante las características del lenguaje de programación o el entorno de tiempo de ejecución, que requieren que los desarrolladores y administradores trabajen juntos.

Es importante entender el flujo de trabajo básico para la mayoría de los lenguajes de programación soportados por OpenShift. El proceso de creación de imágenes de *S2I* está compuesto por dos pasos principales:

- Paso de compilación: Responsable de compilar el código fuente, descargar dependencias de la librería y empaquetar la aplicación como una imagen de contenedor. Además, el paso de compilación envía la imagen al registro de OpenShift para el paso de implementación. Los recursos de OpenShift **BuildConfig (BC)** conducen el paso de compilación.
- Paso de implementación: Responsable de iniciar un pod y hacer que la aplicación esté disponible para OpenShift. Este paso se ejecuta después del paso de compilación, pero solo si el paso de compilación se ejecuta correctamente. Los recursos de OpenShift **DeploymentConfig (DC)** conducen el paso de implementación.

Para el proceso *S2I*, cada aplicación usa sus propios objetos **BuildConfig** y **DeploymentConfig**, cuyo nombre coincide con el nombre de la aplicación. El proceso de implementación se cancela si falla la compilación.

El proceso *S2I* inicia cada paso en un pod separado. El proceso de compilación crea un pod nombrado `<application-name>-build-<number>-<string>`. Para cada intento de compilación, se ejecuta todo el paso de compilación y se guarda un registro. Después de una compilación satisfactoria, la aplicación se inicia en un pod separado, denominado `<application-name>-<string>`.

La consola web de OpenShift se puede usar para acceder a los detalles para cada paso. Para identificar los problemas de compilación, los registros para una compilación pueden evaluarse y analizarse haciendo clic en el enlace **Builds** (Compilaciones) del panel izquierdo, representado de la siguiente manera.

Figura 8.1: Instancias de compilación de un proyecto.

En cada intento de compilación, se proporciona un historial de la compilación, etiquetado con un número, para su evaluación. Al hacer clic en el número de compilación, se muestra la página de detalles de la compilación.

Figura 8.2: Vista detallada de una instancia de compilación

En la pestaña **Logs** (Registros) de la página de detalles de la compilación, se muestra la salida generada por la ejecución de la compilación. Esos registros son útiles para identificar problemas de compilación.

Use el enlace **Deployment Configs** (Configuraciones de implementación) de la sección **Workloads** (Cargas de trabajo) en el panel izquierdo para identificar problemas durante el paso de implementación.

Después de seleccionar la configuración de implementación adecuada, los detalles se muestran en la sección **Details** (Detalles).

La interfaz de la línea de comando **oc** cuenta con varios subcomandos para administrar los registros. Al igual que en la interfaz web, tiene un conjunto de comandos que proporciona información acerca de cada paso. Por ejemplo, para obtener los registros de una configuración de compilación, ejecute el siguiente comando.

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

Si una compilación falla, después de encontrar y solucionar los problemas, ejecute el siguiente comando para solicitar una nueva compilación:

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

Al emitir el comando, OpenShift genera automáticamente un nuevo pod con el proceso de compilación.

Los registros de implementación se pueden verificar con el comando **oc**:

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

Si la implementación se está ejecutando o ha fallado, el comando devuelve los registros del proceso de implementación. De lo contrario, el comando devuelve los registros del pod de la aplicación.

Descripción de problemas comunes

Algunas veces, el código fuente requiere una personalización que puede no estar disponible en entornos contenerizados, como las credenciales de la base de datos, el acceso al sistema de archivos o la información de la cola de mensajes. Esos valores generalmente toman la forma de variables de entorno internas. Los desarrolladores que usan el proceso S2I pueden necesitar tener acceso a esta información.

El comando **oc logs** proporciona información importante acerca de los procesos de compilación, implementación y ejecución de una aplicación durante la ejecución de un pod. Los registros pueden indicar valores que faltan u opciones que deben habilitarse, parámetros o indicadores incorrectos, o incompatibilidades de entorno.



nota

Los registros de aplicación deben estar etiquetados claramente para identificar problemas rápidamente sin la necesidad de conocer las partes internas del contenedor.

Solución de problemas de permisos

OpenShift ejecuta contenedores S2I usando Red Hat Enterprise Linux como la imagen de base y cualquier diferencia de tiempo de ejecución puede causar un error en el proceso S2I. A veces, el desarrollador se encuentra con problemas de permisos, como un acceso negado por permisos inadecuados o permisos de entorno incorrectos establecidos por los administradores. Las imágenes S2I imponen el uso de un usuario diferente al usuario **root** para acceder a sistemas de archivos y recursos externos. Es más, Red Hat Enterprise Linux 7 impone las políticas de SELinux que restringen el acceso a algunos recursos del sistema de archivos, puertos de red o el proceso.

Algunos contenedores pueden requerir un ID de usuario específico, mientras que S2I está diseñado para ejecutar contenedores con un usuario aleatorio, según la política de seguridad predeterminada de OpenShift.

El siguiente Dockerfile crea un contenedor Nexus. Observe la instrucción de **USER** que indica que debe usarse el usuario **nexus**:

```
FROM ubi7/ubi:7.7
...contents omitted...
RUN chown -R nexus:nexus ${NEXUS_HOME}

USER nexus
WORKDIR ${NEXUS_HOME}

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

Intentar usar la imagen generada por este Dockerfile sin direccionar los permisos de volumen genera errores cuando se inicia el contenedor:

```
$ 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...
```

Para solucionar este problema, atenué la seguridad del proyecto OpenShift con el comando **oc adm policy**.

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

Este comando **oc adm policy** permite a OpenShift ejecutar procesos de contenedor con usuarios que no son root. Pero los sistemas de archivos usados en el contenedor también deben estar disponibles para el usuario en ejecución. Esto es especialmente importante cuando el contenedor contiene montajes de volumen.

Para evitar problemas con los permisos del sistema de archivos, las carpetas locales usadas para los montajes de volumen del contenedor deben cumplir con lo siguiente:

- El usuario que ejecuta los procesos de contenedor debe ser el propietario de la carpeta o tener los derechos necesarios. Use el comando **chown** para actualizar la propiedad de la carpeta.
- La carpeta local debe cumplir con los requisitos de SELinux para usarse como un volumen de contenedor. Asigne el grupo **container_file_t** a la carpeta mediante el uso del comando **semanage fcontext -a -t container_file_t <folder>** y, luego, actualice los permisos con el comando **restorecon -R <folder>**.

Solución de problemas con parámetros no válidos

Las aplicaciones con varios contenedores pueden compartir parámetros, como credenciales de inicio de sesión. Asegúrese de que los mismos valores para los parámetros alcancen a todos los contenedores en la aplicación. Por ejemplo, en el caso de una aplicación Python que se ejecuta en un contenedor conectado a otro contenedor que ejecuta una base de datos, asegúrese de que los dos contenedores usen el mismo nombre de usuario y la misma contraseña para la base de datos. Normalmente, los registros del pod de aplicación proporcionan una idea clara de estos problemas y el modo de resolverlos.

Una buena práctica para centralizar los parámetros compartidos es almacenarlos en **ConfigMaps**. Aquellos **ConfigMaps** se pueden insertar a través de **Deployment Config** (Configuración de implementación) en contenedores como variables de entorno. Insertar el mismo **ConfigMap** en diferentes contenedores garantiza que no solo estén disponibles las mismas variables de entorno, sino también los mismos valores. Consulte la siguiente definición de recurso de pod:

```
apiVersion: v1
kind: Pod
...output omitted...
spec:
  containers:
    - name: test-container
  ...output omitted...
  env:
    - name: ENV_1 ①
      valueFrom:
        configMapKeyRef:
          name: configMap_name1
          key: configMap_key_1
  ...output omitted...
  envFrom:
    - configMapRef:
        name: configMap_name_2 ②
  ...output omitted...
```

- ① Se inserta una variable de entorno **ENV_1** en el contenedor. Su valor es el valor de la entrada **configMap_key_1** en el configMap **configMap_name1**.
- ② Todas las entradas en **configMap_name_2** se insertan en el contenedor como variables de entorno con el mismo nombre y los mismos valores.

Solución de problemas con el montaje de volumen

Cuando se vuelve a implementar una aplicación que usa un volumen persistente en un sistema de archivos local, un pod puede tener problemas para asignar una reclamación de volúmenes persistentes incluso si el volumen persistente indica que la reclamación se ha liberado. Para resolver este problema, elimine la reclamación de volumen persistente y, luego, el volumen persistente. A continuación, recrea el volumen persistente.

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

Solución de problemas con imágenes obsoletas

OpenShift extrae imágenes de la fuente indicada en un flujo de imágenes, a menos que ubique una imagen guardada en caché local en el nodo donde el pod está programado para ejecutarse. Si envía una nueva imagen al registro con el mismo nombre y la misma etiqueta, debe eliminar la imagen de cada nodo en el que está programado cada pod con el comando **podman rmi**.

Ejecute el comando **oc adm prune** para obtener una manera automatizada de eliminar imágenes obsoletas y otros recursos.



Referencias

Para obtener información adicional acerca de la solución de problemas de imágenes, consulte la sección *Images* (Imágenes) de la documentación de OpenShift Container Platform, a la que se puede acceder en:

Creación de imágenes

https://docs.openshift.com/container-platform/4.5/openshift_images/create-images.html

La documentación sobre cómo consumir ConfigMap para crear variables de entorno de contenedor se puede encontrar en *Consuming in Environment Variables* (Consumir variables de entorno) del

Configurar un Pod para usar ConfigMaps

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

► Ejercicio Guiado

Solución de problemas de compilación de OpenShift

En este ejercicio, solucionará problemas del proceso de implementación y compilación de OpenShift.

Resultados

Debería poder identificar y resolver los problemas que se presenten durante el proceso de implementación y compilación de una aplicación Node.js.

Andes De Comenzar

Un clúster OpenShift en ejecución.

Obtenga los archivos del trabajo de laboratorio y verifique que Docker y el clúster de OpenShift se estén ejecutando mediante la ejecución del siguiente comando.

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

- ▶ 1. Cargue la configuración de su entorno de aula. Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

- ▶ 2. Ingrese su clon local del repositorio **D0180-apps** de Git y extraiga la bifurcación **master** del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- ▶ 3. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[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. Inicie sesión en OpenShift con el usuario, la contraseña y la URL de la API maestra configurados.

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

You have access to the following projects and can switch between them with 'oc
project <projectname>':
...output omitted...
```

Cree un nuevo proyecto con el nombre **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. Compile una nueva aplicación Node.js con la imagen **Hello World** ubicada en <https://github.com/yourgituser/D0180-apps/> en el directorio **nodejs-helloworld**.
- 5.1. Ejecute el comando **oc new-app** para crear la aplicación Node.js. El comando se proporciona en el archivo [~/D0180/labs/troubleshoot-s2i/command.txt](#).

```
[student@workstation D0180-apps]$ oc new-app --as-deployment-config \
> --context-dir=nodejs-helloworld \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#troubleshoot-s2i \
> -i nodejs:12 --name nodejs-hello --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/npm-proxy
--> Found image a2b5ec2 ...output omitted...

  Node.js 12
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "nodejs-hello" created
  buildconfig.build.openshift.io "nodejs-hello" created
  deploymentconfig.apps.openshift.io "nodejs-hello" created
  service "nodejs-hello" created
--> Success
  Build scheduled, use 'oc logs -f bc/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.
```

El **-i** indica la imagen del compilador que debe usarse, **nodejs:12** en este caso.

La opción **--context-dir** define la carpeta dentro del proyecto que contiene el código fuente de la aplicación que se debe compilar.

La opción **--build-env** define una variable de entorno al pod de constructor. En este caso, proporciona la variable de entorno **npm_config_registry** al pod de compilador, para que pueda llegar al registro de NPM.

**Importante**

En el comando anterior, no debe haber espacios entre **registry=** y la URL del servidor Nexus.

- 5.2. Espere hasta que la aplicación finalice la compilación mediante el monitoreo del progreso con el comando **oc get pods -w**. El pod pasa de un estado de **ejecución** a **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
```

El proceso de compilación falla y, por lo tanto, no se está ejecutando ninguna aplicación. Las fallas de compilación generalmente son consecuencia de errores de sintaxis en el código fuente o de dependencias que faltan. En el siguiente paso, se investigan las causas específicas de esta falla.

- 5.3. Evalúe los errores que surgieron durante el proceso de compilación.

La compilación es desencadenada por la configuración de compilación (**bc**), creada por OpenShift cuando se inicia el proceso de S2I. De forma predeterminada, el proceso S2I de OpenShift crea una configuración de compilación con el nombre dado: **nodejs-hello**, que es responsable de desencadenar el proceso de compilación.

Ejecute el comando **oc** con el subcomando **logs** en una ventana de terminal para revisar la salida del proceso de compilación:

```
[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 'nodejs-helloworld'
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
subprocess exited with status 1
...output omitted...
```

El registro muestra un error que ocurrió durante el proceso de compilación. En esta salida, se indica que no hay una versión compatible para la dependencia **express**. Sin embargo, el motivo es que el formato usado por la dependencia express no es válido.

► 6. Actualice el proceso de compilación para el proyecto.

El desarrollador usa una versión no estándar del framework (marco) Express, que está disponible a nivel local en cada estación de trabajo del desarrollador. Debido a los estándares de la empresa, la versión debe descargarse del registro oficial Node.js y, desde la entrada del desarrollador, es compatible con la versión 4.14.x.

6.1. Corrija el archivo **package.json**.

Use su editor preferido para abrir el archivo **~/D0180-apps/nodejs-helloworld/package.json**. Revise las versiones de las dependencias provistas por los desarrolladores. Usa una versión incorrecta de la dependencia Express, que no es compatible con la versión soportada provista por la empresa (~4.14.2). Actualice la versión de la dependencia de la siguiente manera.

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



nota

Observe la **x** en la versión. Indica que se debe usar la versión más alta, pero la versión debe comenzar con **4.14..**

6.2. Confirme y envíe los cambios realizados en el proyecto.

En la ventana de terminal, ejecute el siguiente comando para confirmar y enviar los cambios:

```
[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. Vuelva a implementar el proceso S2I.

7.1. Para reiniciar el paso de compilación, ejecute el siguiente comando:

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

El paso de compilación se reinicia, y se crea un nuevo pod de compilación. Verifique el registro ejecutando el comando **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
```

La compilación se realizó correctamente; no obstante, esto no indica que se haya iniciado la aplicación.

- 7.2. Evalúe el estado del proceso de compilación actual. Ejecute el comando **oc get pods** para verificar el estado de la aplicación Node.js.

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

Según la siguiente salida, la segunda compilación se completó, pero la aplicación se encuentra en estado de error.

NAME	READY	STATUS	RESTARTS	AGE
nodejs-hello-1-build	0/1	Error	0	29m
nodejs-hello-1-rpx1d	0/1	CrashLoopBackOff	6	6m
nodejs-hello-2-build	0/1	Completed	0	7m

El nombre del pod de la aplicación (**nodejs-hello-1-rpx1d**) se genera aleatoriamente y puede diferir del suyo.

- 7.3. Consulte los registros generados por el pod de la aplicación.

```
[student@workstation D0180-apps]$ oc logs dc/nodejs-hello
...output omitted...
npm info using npm@6.14.5
npm info using node@v12.18.2
npm ERR! missing script: start
...output omitted...
```



nota

El comando **oc logs dc/nodejs-hello** descarga los registros del pod de implementación. En el caso de una implementación correcta, ese comando descarga los registros del pod de la aplicación, como se mostró anteriormente.

La aplicación no se puede iniciar porque falta la declaración del script de inicio.

- 8. Solucione el problema actualizando el pod de la aplicación.

- 8.1. Actualice el archivo **package.json** para definir un comando de inicio.

La salida anterior indica que falta el atributo **start** en el archivo **~/D0180-apps/nodejs-helloworld/package.json** en el campo **scripts**. El atributo **start** define un comando para ejecutar cuando se inicia la aplicación. Invoca el binario **node** que ejecuta la aplicación **app.js**.

Para solucionar el problema, agregue al archivo **package.json** el siguiente atributo. No se olvide de la coma después del corchete.

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

```

8.2. Confirme y envíe los cambios realizados en el proyecto:

```
[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
```

Continúe el paso de implementación desde el proceso S2I.

8.3. Reinicie el paso de compilación.

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

8.4. Evalúe el estado del proceso de compilación actual. Ejecute el comando para consultar el estado de la aplicación Node.js. Espere que finalice la compilación más reciente.

```
[student@workstation D0180-apps]$ oc get pods -w
NAME          READY   STATUS      RESTARTS   AGE
nodejs-hello-1-build  0/1     Error       0          66m
nodejs-hello-1-mtxsh  0/1     CrashLoopBackOff  9          23m
nodejs-hello-2-build  0/1     Completed    0          28m
nodejs-hello-3-build  1/1     Running     0          2m3s
nodejs-hello-2-deploy 0/1     Pending      0          0s
nodejs-hello-2-deploy 0/1     Pending      0          0s
nodejs-hello-2-deploy 0/1     ContainerCreating  0          0s
nodejs-hello-3-build 0/1 Completed 0 3m9s
nodejs-hello-2-deploy 1/1     Running     0          4s
nodejs-hello-2-8tsl4   0/1     Pending      0          0s
nodejs-hello-2-8tsl4   0/1     Pending      0          1s
nodejs-hello-2-8tsl4   0/1     ContainerCreating  0          1s
nodejs-hello-2-8tsl4 1/1 Running 0 50s
nodejs-hello-1-mtxsh  0/1     Terminating  9          25m
nodejs-hello-1-mtxsh  0/1     Terminating  9          25m
nodejs-hello-2-deploy 0/1 Completed 0 61s
nodejs-hello-2-deploy 0/1     Terminating  0          61s
```

```
nodejs-hello-2-deploy  0/1  Terminating  0   61s
nodejs-hello-1-mtxsh  0/1  Terminating  9   25m
nodejs-hello-1-mtxsh  0/1  Terminating  9   25m
```

De acuerdo con la salida, la compilación se realizó correctamente y la aplicación se puede iniciar sin errores. La salida también proporciona información sobre cómo se creó el pod de implementación (**nodejs-hello-2-deploy**) e indica que se completó correctamente y finalizó. Cuando se pone a disposición el nuevo pod de la aplicación (**nodejs-hello-2-8ts14**), se descarta el anterior (**nodejs-hello-1-mtxsh**).

- 8.5. Consulte los registros generados por el pod de la aplicación **nodejs-hello**.

```
[student@workstation D0180-apps]$ oc logs dc/nodejs-hello
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@2.15.1
npm info using node@v4.6.2
npm info prestart nodejs-helloworld@1.0.0
npm info start nodejs-helloworld@1.0.0

> nodejs-helloworld@1.0.0 start /opt/app-root/src
> node app.js

Example app listening on port 8080!
```

La aplicación ahora se está ejecutando en el puerto 8080.

► 9. Pruebe la aplicación.

- 9.1. Ejecute el comando **oc** con el subcomando **expose** para exponer la aplicación:

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

- 9.2. Recupere la dirección relacionada con la aplicación.

```
[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-nodejs.apps.cluster.lab.example.com
  port:
    targetPort: 8080-tcp
  to:
```

```
kind: Service
name: do180-apps
...output omitted...
```

- 9.3. Acceda a la aplicación desde la máquina virtual **workstation** mediante el comando **curl**:

```
[student@workstation DO180-apps]$ curl -w "\n" \
> http://nodejs-hello-nodejs.apps.cluster.lab.example.com
Hello world!
```

En la salida, se muestra que la aplicación está en funcionamiento.

Finalizar

En **workstation**, ejecute el script **lab troubleshoot-s2i finish** para terminar este ejercicio.

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

Esto concluye el ejercicio.

Solución de problemas de aplicaciones contenerizadas

Objetivos

Tras finalizar esta sección, usted deberá ser capaz de realizar lo siguiente:

- Implementar técnicas para solucionar problemas y depurar las aplicaciones en contenedores.
- Usar la característica de reenvío de puertos de la herramienta de cliente de OpenShift.
- Ver los registros del contenedor.
- Ver los eventos del clúster de OpenShift.

Reenvío de puertos para la solución de problemas

Algunas veces, los desarrolladores y los administradores de sistemas necesitan acceso de red especial a un contenedor que no necesitarían los usuarios de la aplicación. Por ejemplo, los desarrolladores pueden necesitar usar la consola de administración para un servicio de mensajería o base de datos, o los administradores del sistema pueden usar el acceso de SSH a un contenedor para reiniciar un servicio terminado. Este acceso a la red, en forma de puertos de red, generalmente no está expuesto por las configuraciones predeterminadas del contenedor y tiende a requerir clientes especializados usados por los desarrolladores y administradores del sistema.

Podman ofrece funciones de reenvío de puertos mediante el uso de la opción **-p** junto con el subcomando **run**. En este caso, no existe distinción entre el acceso de red para el acceso a aplicaciones regulares y para solución de problemas. Como repaso, el siguiente es un ejemplo de configuración de un reenvío de puertos mediante la asignación del puerto desde el host hasta un servidor de bases de datos que se ejecuta dentro de un contenedor:

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

El comando anterior asigna el puerto de host 30306 hasta el puerto 3306 en el contenedor **db**. Este contenedor se creó a partir de la imagen **mysql**, que inicia un servidor MySQL que escucha en el puerto 3306.

OpenShift proporciona el comando **oc port-forward** que reenvía un puerto local a un puerto de pod. Es diferente a tener acceso a un pod a través de un recurso de servicio:

- La asignación de reenvío de puertos existe solo en la estación de trabajo donde se ejecuta el cliente **oc**, mientras que un servicio asigna un puerto a todos los usuarios de red.
- Un servicio balancea la carga de conexiones para potencialmente varios pods, mientras que una asignación de reenvío de puertos reenvía conexiones a un solo pod.

Este es un ejemplo del comando **oc port-forward**:

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

El comando anterior envía el puerto 30306 de la máquina del desarrollador al puerto 3306 en el pod **db**, donde un servidor MySQL (dentro de un contenedor) acepta las conexiones de red.

**nota**

Al ejecutar este comando, asegúrese de dejar la ventana de terminal en ejecución. Si cierra la ventana o cancela el proceso, se detendrá la asignación de puertos.

Si bien el método de asignación (reenvío de puertos) **podman run -p** solo puede configurarse cuando se inicia el contenedor, la asignación con el comando **oc port-forward** puede crearse y destruirse en cualquier momento después de la creación de un pod.

**nota**

La creación de un servicio del tipo **NodePort** para un pod de base de datos sería similar a ejecutar **podman run -p**. No obstante, Red Hat desalienta el uso del enfoque **NodePort** para evitar exponer el servicio a conexiones directas. Una asignación como reenvío de puertos en OpenShift se considera una alternativa más segura.

Habilitación de la depuración remota con reenvío de puertos

Otro uso para la característica de reenvío de puertos es habilitar la depuración remota. Muchos *entornos de desarrollo integrados (IDE)* proporcionan la capacidad de depurar un aplicación de forma remota.

Por ejemplo, JBoss Developer Studio (JBDS) permite a los usuarios usar Java Debug Wire Protocol (JDWP) para comunicarse entre un depurador (JBDS) y la máquina virtual Java. Cuando se habilita, los desarrolladores pueden ir paso a paso en cada línea de código, ya que se ejecuta en tiempo real.

Para que JDWP funcione, debe iniciarse la máquina virtual Java (JVM) donde se ejecuta la aplicación con opciones que habiliten la depuración remota. Por ejemplo, los usuarios de WildFly y JBoss EAP deben configurar estas opciones en el momento del inicio de servidor de aplicaciones. La siguiente línea en el archivo **standalone.conf** habilita la depuración remota mediante la apertura del puerto TCP 8787 de JDWP para una instancia WildFly o EAP que se ejecuta en modo independiente (standalone):

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

Una vez que se inicia el servidor con el depurador que escucha en el puerto 8787, se debe crear una asignación de reenvío de puertos para reenviar las conexiones desde un puerto TCP local no usado hasta el puerto 8787 en el pod de EAP. Si la estación de trabajo del desarrollador no tiene ninguna JVM local ejecutándose con depuración remota habilitada, el puerto local también puede ser 8787.

El siguiente comando asume un pod WildFly con el nombre **jappserver** que ejecuta un contenedor a partir de una imagen configurada previamente para habilitar la depuración remota:

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

Una vez que se habilite el depurador y se cree la asignación de reenvío de puertos, los usuarios pueden establecer puntos de interrupción en su IDE de preferencia y ejecutar el depurador señalando el nombre de host y el puerto de depuración de la aplicación (en esta instancia, 8787).

Acceso a los registros del contenedor

Podman y OpenShift proporcionan la capacidad de ver registros en contenedores en ejecución y pods para facilitar la solución de problemas. Sin embargo, ninguno de ellos conoce los registros específicos de la aplicación. Ambos esperan que la aplicación se configure para enviar todas las salidas de los registros a la salida estándar.

Un contenedor es solo un árbol de procesos desde el punto de vista del SO del host. Cuando Podman inicia un contenedor, ya sea directamente o en el clúster de RHOPC, redirige la salida estándar y el error estándar del contenedor, y los guarda en el disco como parte del almacenamiento efímero del contenedor. De este modo, los registros del contenedor pueden verse con los comandos **podman** y **oc**, aun después de que se detuvo el contenedor, pero no se eliminó.

Para obtener la salida de un contenedor en ejecución, use el siguiente comando **podman**:

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

En OpenShift, el siguiente comando muestra la salida para un contenedor dentro de un pod:

```
$ oc logs <podName> [-c <containerName>]
```



nota

El nombre del contenedor es opcional si existe solo un contenedor, ya que **oc** usará el único contenedor en ejecución de forma predeterminada y regresará a la salida.

Eventos de OpenShift

Algunos desarrolladores consideran que los registros de Podman y OpenShift tienen un nivel demasiado bajo, lo que dificulta la solución de problemas. Afortunadamente, OpenShift proporciona un registro de alto nivel y la utilidad de auditoría denominada **events** (eventos).

Los eventos de OpenShift señalan acciones importantes, como iniciar un contenedor o destruir un pod.

Para leer eventos de OpenShift, use el subcomando **get** con el tipo de recurso **events** para el comando **oc**, de la siguiente manera.

```
$ oc get events
```

Los eventos enumerados por el comando **oc** de esta manera no se filtran y abarcan todo el clúster de RHOPC. Una canalización a filtros UNIX estándares, como **grep**, puede ser útil, pero OpenShift ofrece una alternativa para consultar eventos de clúster. El enfoque es provisto por el subcomando **describe**.

Por ejemplo, para solo recuperar los eventos que se relacionan con el pod **mysql**, consulte el campo **Events** (Eventos) de la salida del comando **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...
```

Acceso a los contenedores en ejecución

Los comandos **podman logs** y **oc logs** pueden ser útiles para ver la salida enviada por cualquier contenedor. No obstante, la salida no necesariamente muestra toda la información disponible si la aplicación se configura para enviar registros a un archivo. Otros escenarios de solución de problemas pueden requerir una inspección del entorno de contenedores como lo ven los procesos dentro del contenedor, como verificar la conectividad externa.

Como solución, Podman y OpenShift proporcionan el subcomando **exec**, lo que permite la creación de nuevos procesos dentro de un contenedor en ejecución y reenvía la entrada y la salida estándares de estos procesos al terminal del usuario. En la siguiente pantalla, se muestra el uso del comando **podman exec**:

```
$ sudo podman exec [options] container command [arguments]
```

La sintaxis general del comando **oc exec** es:

```
$ oc exec [options] pod [-c container] -- command [arguments]
```

Para ejecutar un solo comando interactivo o iniciar una shell, agregue las opciones **-it**. En el siguiente ejemplo, se inicia una shell de Bash para el pod **myhttpd**:

```
$ oc exec -it myhttpd /bin/bash
```

Puede usar este comando para acceder a los registros de la aplicación guardados en el disco (como parte del almacenamiento efímero del contenedor). Por ejemplo, para mostrar el registro de errores de Apache desde un contenedor, ejecute el siguiente comando:

```
$ sudo podman exec apache-container cat /var/log/httpd/error_log
```

Reemplazo de binarios del contenedor

Muchas imágenes de contenedores no contienen todos los comandos de solución de problemas que los usuarios esperan encontrar en las instalaciones de SO regulares, como **telnet**, **netcat**, **ip** o **traceroute**. Eliminar utilidades o binarios básicos de una imagen permite que la imagen permanezca ligera, ejecutando muchos contenedores por host.

Una forma de acceder de manera temporal a algunos de estos comandos que faltan es montar las carpetas de binarios del host, como **/bin**, **/sbin** y **/lib**, como volúmenes dentro del contenedor. Esto es posible porque la opción **-v** del comando **podman run** no requiere la presentación de instrucciones **VOLUME** que coincidan en el **Dockerfile** del contenedor de imagen.

**nota**

Para acceder a estos comandos en OpenShift, necesita cambiar la definición del recurso de pod para definir los objetos **volumeMounts** y **volumeClaims**. También necesita crear un volumen **hostPath** persistente.

El siguiente comando inicia un contenedor y reemplaza la carpeta **/bin** de imágenes con la del host. También inicia una shell interactiva dentro del contenedor.

```
$ sudo podman run -it -v /bin:/bin image /bin/bash
```

**nota**

El directorio de binarios que debe reemplazar depende de la imagen base del SO. Por ejemplo, algunos comandos requieren bibliotecas compartidas desde el directorio **/lib**. Algunas distribuciones de Linux tienen diferentes contenidos en **/bin**, **/usr/bin**, **/lib** o **/usr/lib**, lo que requerirá el uso de la opción **-v** para cada directorio.

Como alternativa, puede incluir estas utilidades en la imagen de base. Para ello, agregue instrucciones en una definición de compilación **Dockerfile**. Por ejemplo, examine el siguiente extracto de una definición de **Dockerfile**, que es un elemento secundario de la imagen **rhel7.5** usado en todo el curso. La instrucción **RUN** instala las herramientas que se usan generalmente para solucionar problemas de redes.

```
FROM ubi7/ubi:7.7

RUN yum install -y \
    less \
    dig \
    ping \
    iputils && \
    yum clean all
```

Cuando se compila la imagen y se crea el contenedor, será idéntico a una imagen de contenedor **rhel7.5**, más las herramientas adicionales disponibles.

Transferencia de archivos hacia y desde los contenedores

Cuando soluciona problemas o administra una aplicación, puede que necesite obtener o transferir archivos hacia y desde los contenedores en ejecución, como archivos de configuración o de registro. Hay diversas maneras de mover archivos hacia y desde los contenedores, como se describe en la siguiente lista.

Montajes de volúmenes

Otra opción para copiar archivos desde el host a un contenedor es usar montajes de volúmenes. Puede montar un directorio local para copiar datos en un contenedor. Por ejemplo, el siguiente comando establece el directorio **/conf** del host como volumen para usar para el directorio de configuración de Apache en el contenedor. Esto crea una manera simple de administrar el servidor de Apache sin tener que volver a compilar la imagen de contenedor.

```
$ sudo podman run -v /conf:/etc/httpd/conf -d do180/apache
```

podman cp

El subcomando **cp** permite a los usuarios copiar archivos tanto para introducirlos en un contenedor en ejecución como para extraerlos de este. Para copiar un archivo en un contenedor con el nombre **todoapi**, ejecute el siguiente comando:

```
$ sudo podman cp standalone.conf todoapi:/opt/jboss/standalone/conf/
standalone.conf
```

Para copiar un archivo del contenedor al host, cambie el orden del comando anterior:

```
$ sudo podman cp todoapi:/opt/jboss/standalone/conf/standalone.conf .
```

El comando **podman cp** tiene la ventaja de trabajar con contenedores que ya se iniciaron, mientras que la siguiente alternativa (montajes de volúmenes) requiere cambios al comando usado para iniciar un contenedor.

podman exec

Para los contenedores que ya se ejecutaron, el comando **podman exec** puede crear una tubería para pasar archivos hacia y desde el contenedor en ejecución mediante el anexo de comandos que se ejecutan en el contenedor. En el siguiente ejemplo se muestra cómo pasar y ejecutar un archivo SQL dentro de un contenedor de MySQL:

```
$ sudo podman exec -i <container> mysql -uroot -proot < /path/on/host/db.sql <
db.sql
```

Con el mismo concepto, es posible extraer datos de un contenedor en ejecución y colocarlos en la máquina del host. Un ejemplo útil de esto es el uso de la utilidad **mysqldump**, la cual crea una copia de seguridad de la base de datos MySQL del contenedor y la coloca en el host.

```
$ sudo 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
```

El comando anterior usa las variables de entorno del contenedor para conectarse con el servidor de MySQL y ejecutar el comando **mysqldump**; además, redirige la salida a un archivo en la máquina del host. Supone que la imagen de contenedor proporciona la utilidad **mysqldump**, de modo que no hace falta instalar las herramientas de administración de MySQL en el host.

El comando **oc rsync** proporciona una funcionalidad similar a **podman cp** para contenedores que se ejecutan en pods de OpenShift.



Referencias

Para obtener información adicional acerca del reenvío de puertos, consulte la sección *Reenvío de puertos* de la documentación de OpenShift Container Platform en

Arquitectura

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/architecture/index/

Para obtener más información acerca de los comandos de CLI para el reenvío de puertos, consulte el capítulo *Reenvío de puertos* de la documentación de OpenShift Container Platform en

Desarrollo de aplicaciones

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/applications/index/

► Ejercicio Guiado

Configuración de los registros del contenedor de Apache para depuración

En este ejercicio, configurará un contenedor httpd de Apache para enviar los registros a **stdout**; luego, verificará los eventos y los registros de Podman.

Resultados

Debería poder configurar un contenedor httpd de Apache para enviar registros de depuración a **stdout** y visualizarlos con el comando **podman logs**.

Antes De Comenzar

Un clúster OpenShift en ejecución.

Obtenga los archivos del trabajo de laboratorio y verifique que Docker y el clúster de OpenShift se estén ejecutando mediante la ejecución del siguiente comando.

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

- 1. Configure un servidor web Apache para enviar mensajes de registro a la salida estándar y actualice el nivel de registro predeterminado.

- 1.1. El nivel de registro predeterminado para la imagen httpd de Apache es **warn**. Cambie el nivel de registro predeterminado para el contenedor a **debug** y redirija los mensajes de registro a **stdout** reemplazando el archivo de configuración **httpd.conf** predeterminado. Para ello, cree una imagen personalizada desde la máquina virtual **workstation**.

Revise brevemente el archivo **httpd.conf** personalizado que se encuentra en **/home/student/D0180/labs/troubleshoot-container/conf/httpd.conf**.

- Observe la directiva **ErrorLog** en el archivo:

```
ErrorLog /dev/stdout
```

La directiva envía los mensajes de registro de errores httpd a la salida estándar del contenedor.

- Observe la directiva **LogLevel** en el archivo.

```
LogLevel debug
```

La directiva cambia el nivel de registro predeterminado a **debug**.

- Observe la directiva **CustomLog** en el archivo.

```
CustomLog /dev/stdout common
```

La directiva redirige los mensajes de registro de acceso httpd a la salida estándar del contenedor.

- ▶ 2. Compile un contenedor personalizado para guardar un archivo de configuración actualizado al contenedor.
- 2.1. En la ventana de terminal, ejecute los siguientes comandos para compilar una nueva imagen.

```
[student@workstation ~]$ cd ~/DO180/labs/troubleshoot-container
[student@workstation troubleshoot-container]$ sudo podman build \
> -t troubleshoot-container .
STEP 1: FROM redhattraining/httpd-parent
...output omitted...
--> e23d...c1de
STEP 7: COMMIT troubleshoot-container
[student@workstation troubleshoot-container]$ cd ~
```

2.2. Verifique que se haya creado la imagen.

```
[student@workstation ~]$ sudo podman images
```

La nueva imagen debe estar disponible en el almacenamiento 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. Cree un nuevo contenedor httpd desde la imagen personalizada.

```
[student@workstation ~]$ sudo podman run \
> --name troubleshoot-container -d \
> -p 10080:80 troubleshoot-container
4c8bb12815cc02f4eef0254632b7179bd5ce230d83373b49761b1ac41fc067a9
```

- ▶ 4. Revise los mensajes y eventos del registro del contenedor.

4.1. Vea los mensajes del registro de depuración desde el contenedor con el comando **podman logs**:

```
[student@workstation ~]$ sudo podman logs -f troubleshoot-container
... [mpm_event:notice] [pid 1:tid...] AH00489: Apache/2.4.25 (Unix) configur...
... [mpm_event:info] [pid 1:tid...] AH00490: Server built: Mar 21 2017 20:50:17
... [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
```

Observe los registros de depuración, disponibles en la salida estándar.

- 4.2. Abra un nuevo terminal y acceda a la página de inicio del servidor web mediante el comando **curl**:

```
[student@workstation ~]$ curl http://127.0.0.1:10080
Hello from the httpd-parent container!
```

- 4.3. Revise las nuevas entradas en el registro. Observe en el terminal que está en ejecución el comando **podman logs** para ver las nuevas entradas.

```
[student@workstation ~]$ sudo podman logs troubleshoot-container
...[authz_core:debug] ...: authorization result of Require all granted: granted
...[authz_core:debug] ...: authorization result of <RequireAny>: granted
10.88.0.1 - - [08/Mar/2019:20:30:53 +0000] "GET / HTTP/1.1" 200 45
```

- 4.4. Detenga el comando de Podman con **Ctrl+C**.

Finalizar

En **workstation**, ejecute el script **lab troubleshoot-container finish** para terminar este trabajo de laboratorio.

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

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Solución de problemas de aplicaciones en contenedores

Lista de verificación de rendimiento

En este trabajo de laboratorio, solucionará problemas del proceso de implementación y compilación de OpenShift para una aplicación Node.js.

Resultados

Debería poder identificar y resolver los problemas que se presenten durante el proceso de implementación y compilación de una aplicación Node.js.

Andes De Comenzar

Un clúster OpenShift en ejecución.

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Cargue la configuración de su entorno de aula. Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

2. Ingrese su clon local del repositorio **D0180-apps** de Git y extraiga la bifurcación **master** del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

3. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[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. Inicie sesión en OpenShift con el usuario, la contraseña y la URL de la API maestra configurados.
5. Cree un nuevo proyecto con el nombre **youruser-nodejs-app**:

6. En el proyecto de OpenShift **youruser-nodejs-app**, cree una nueva aplicación a partir del código fuente ubicado en el directorio de **nodejs-app** en el repositorio de Git en <https://github.com/yourgituser/D0180-apps>. Use el nombre **nodejs-dev** para la aplicación.
Espere que el proceso de compilación de la aplicación falle. Monitoree el proceso de compilación e identifique la falla de compilación.
7. Actualice la versión de la dependencia **express** en el archivo **package.json** con un valor de **4.x**. Confirme y envíe los cambios al repositorio Git.
8. Vuelva a compilar la aplicación. Verifique que la aplicación se compile sin errores.
9. Verifique que la aplicación no se esté ejecutando debido a un error de tiempo de ejecución. Revise los registros e identifique el problema.
10. Corrija la ortografía de la dependencia en la primera línea del archivo **server.js**. Confirme y envíe los cambios de la aplicación al repositorio Git. Vuelva a compilar la aplicación. Después de que se compila la aplicación, verifique que la aplicación se esté ejecutando.
11. Cree una ruta para la aplicación y pruebe el acceso a la aplicación. Espere recibir un mensaje de error. Consulte los registros para identificar el error.
12. Reemplace **process.environment** con **process.env** en el archivo **server.js** para corregir el error. Confirme y envíe los cambios de la aplicación al repositorio Git. Vuelva a compilar la aplicación. Cuando se implemente la nueva aplicación, verifique que la aplicación no genere errores cuando acceda a la URL de la aplicación.

Evaluación

Evalúe su trabajo ejecutando el comando **lab troubleshoot-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

Finalizar

En **workstation**, ejecute el comando **lab troubleshoot-review finish** para terminar este trabajo de laboratorio.

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

Esto concluye el trabajo de laboratorio.

► Solución

Solución de problemas de aplicaciones en contenedores

Lista de verificación de rendimiento

En este trabajo de laboratorio, solucionará problemas del proceso de implementación y compilación de OpenShift para una aplicación Node.js.

Resultados

Debería poder identificar y resolver los problemas que se presenten durante el proceso de implementación y compilación de una aplicación Node.js.

Andes De Comenzar

Un clúster OpenShift en ejecución.

Abra un terminal en **workstation** como usuario **student** y ejecute el siguiente comando:

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

1. Cargue la configuración de su entorno de aula. Ejecute el siguiente comando para cargar las variables de entorno creadas en el primer ejercicio guiado:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```

2. Ingrese su clon local del repositorio **D0180-apps** de Git y extraiga la bifurcación **master** del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

3. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[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. Inicie sesión en OpenShift con el usuario, la contraseña y la URL de la API maestra configurados.

```
[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. Cree un nuevo proyecto con el nombre **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. En el proyecto de OpenShift **youruser-nodejs-app**, cree una nueva aplicación a partir del código fuente ubicado en el directorio de **nodejs-app** en el repositorio de Git en <https://github.com/yourgituser/D0180-apps>. Use el nombre **nodejs-dev** para la aplicación.

Espere que el proceso de compilación de la aplicación falle. Monitoree el proceso de compilación e identifique la falla de compilación.

- 6.1. Ejecute el comando **oc new-app** para crear la aplicación Node.js.

```
[student@workstation ~]$ oc new-app --as-deployment-config --context-dir=nodejs-
app \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#troubleshoot-review \
> -i nodejs:12 --name nodejs-dev --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/npm-proxy
--> Found image a2b5ec2 ...output omitted...

Node.js 12
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "nodejs-dev" created
buildconfig.build.openshift.io "nodejs-dev" created
deploymentconfig.apps.openshift.io "nodejs-dev" created
service "nodejs-dev" created
--> Success
Build scheduled, use 'oc logs -f bc/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 svc/nodejs-dev'
Run 'oc status' to view your app.
```

- 6.2. Monitoree el progreso de la compilación con el comando **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 'nodejs-app'
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
```

El proceso de compilación falla y, por lo tanto, no se está ejecutando ninguna aplicación. El registro de compilación indica que no hay una versión del paquete **express** que coincida con una especificación de versión de **4.20.x**.

- 6.3. Use el comando **oc get pods** para confirmar que la aplicación no está implementada:

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
nodejs-dev-1-build   0/1     Error      0          2m
```

7. Actualice la versión de la dependencia **express** en el archivo **package.json** con un valor de **4.x**. Confirme y envíe los cambios al repositorio Git.
- 7.1. Edite el archivo **package.json** en el subdirectorio **nodejs-app** y cambie la versión de la dependencia **express** a **4.x**. Guarde el archivo.

```
[student@workstation DO180-apps]$ cd nodejs-app
```

```
[student@workstation nodejs-app]$ sed -i s/4.20/4.x/ package.json
```

El archivo cuenta con el siguiente contenido:

```
[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"
  }
}
```

- 7.2. Confirme y envíe los cambios realizados en el proyecto.

En la ventana de terminal, ejecute el siguiente comando para confirmar y enviar los cambios:

```
[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. Vuelva a compilar la aplicación. Verifique que la aplicación se compile sin errores.

- 8.1. Use el comando **oc start-build** para volver a compilar la aplicación.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-2 started
```

- 8.2. Use el comando **oc logs** para monitorear los registros del proceso de compilación:

```
[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 compilación se realiza correctamente si se envía una imagen al registro interno de OpenShift.

9. Verifique que la aplicación no se esté ejecutando debido a un error de tiempo de ejecución. Revise los registros e identifique el problema.

- 9.1. Use el comando **oc get pods** para comprobar el estado de la implementación del pod de la aplicación. Finalmente, verá que la primera implementación de la aplicación tiene un estado de **CrashLoopBackoff**.

```
[student@workstation nodejs-app]$ oc get pods
NAME          READY   STATUS      RESTARTS   AGE
nodejs-dev-1-86gg5  0/1     CrashLoopBackOff  6          7m
nodejs-dev-1-build  0/1     Error       0          26m
nodejs-dev-2-build  0/1     Completed    0          11m
```

- 9.2. Use el comando **oc logs -f dc/nodejs-dev** para seguir los registros de la implementación de la aplicación:

```
[student@workstation nodejs-app]$ oc logs -f dc/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!
npm ERR! Failed at the nodejs-app@1.0.0 start script 'node server.js'.
npm ERR! This is most likely a problem with the nodejs-app package,
npm ERR! not with npm itself.
...output omitted...
```

El registro indica que el archivo **server.js** intenta cargar un módulo denominado **http-error**. La variable **dependencies** del archivo **packages** indica que el nombre del módulo es **html-errors**, no **http-error**.

10. Corrija la ortografía de la dependencia en la primera línea del archivo **server.js**. Confirme y envíe los cambios de la aplicación al repositorio Git. Vuelva a compilar la aplicación. Después de que se compila la aplicación, verifique que la aplicación se esté ejecutando.
 - 10.1. Corrija la ortografía del módulo en la primera línea de **server.js** de **http-error** a **html-errors**. Guarde el archivo.

```
[student@workstation nodejs-app]$ sed -i s/http-error/html-errors/ server.js
```

El archivo cuenta con el siguiente contenido:

```
[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. Confirme y envíe los cambios realizados en el proyecto.

```
[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. Use el comando **oc start-build** para volver a compilar la aplicación.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-3 started
```

- 10.4. Use el comando **oc logs** para monitorear los registros del proceso de compilación:

```
[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
```

- 10.5. Use el comando **oc get pods -w** para monitorear la implementación de los pods para la aplicación **nodejs-dev**:

```
[student@workstation nodejs-app]$ oc get pods -w
NAME          READY   STATUS    RESTARTS   AGE
nodejs-dev-1-build  0/1     Error      0          6h9m
nodejs-dev-2-build  0/1     Completed   0          5h55m
nodejs-dev-2-xt8q4  1/1     Running   0          4m
nodejs-dev-3-build  0/1     Completed   0          7m57s
```

Después de una tercera compilación, la segunda implementación genera un estado de **Running** (En ejecución).

11. Cree una ruta para la aplicación y pruebe el acceso a la aplicación. Espere recibir un mensaje de error. Consulte los registros para identificar el error.

- 11.1. Use el comando **oc expose** para crear una ruta para la aplicación **nodejs-dev**:

```
[student@workstation nodejs-app]$ oc expose svc nodejs-dev
route.route.openshift.io/nodejs-dev exposed
```

- 11.2. Use el comando **oc get route** para recuperar la URL de la ruta **nodejs-dev**:

```
[student@workstation nodejs-app]$ oc get route
NAME          HOST/PORT
nodejs-dev    nodejs-dev-your_user-nodejs-app.wildcard_domain ...
```

- 11.3. Use **curl** para acceder a la ruta. Espere recibir un mensaje de error.

```
[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. Consulte los registros para la configuración de la implementación de **nodejs-dev**:

```
[student@workstation nodejs-app]$ oc logs dc/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@2.15.1
npm info using node@v4.6.2
npm info prestart nodejs-app@1.0.0
npm info start nodejs-app@1.0.0

> nodejs-app@1.0.0 start /opt/app-root/src
> node server.js

Example app listening on port 8080!
TypeError: Cannot read property 'HOSTNAME' of undefined
...output omitted...
```

La sección correspondiente del archivo **server.js** es:

```
app.get('/', function (req, res) {
  res.send('Hello World from pod: ' + process.env.HOSTNAME + '\n')
});
```

Un objeto **process** en Node.js contiene una referencia a un objeto **env**, no a un objeto **environment**.

12. Reemplace **process.environment** con **process.env** en el archivo **server.js** para corregir el error. Confirme y envíe los cambios de la aplicación al repositorio Git. Vuelva a compilar la aplicación. Cuando se implemente la nueva aplicación, verifique que la aplicación no genere errores cuando acceda a la URL de la aplicación.

- 12.1. Reemplace **process.environment** con **process.env** en el archivo **server.js** para corregir el error.

```
[student@workstation nodejs-app]$ sed -i \
> s/process.environment/process.env/ server.js
```

El archivo cuenta con el siguiente contenido:

```
[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. Confirme y envíe los cambios realizados en el proyecto.

```
[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. Use el comando **oc start-build** para volver a compilar la aplicación.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-4 started
```

12.4. Use el comando **oc logs** para monitorear los registros del proceso de compilación:

```
[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. Use el comando **oc get pods** para monitorear la implementación de los pods para la aplicación **nodejs-dev**:

```
[student@workstation nodejs-app]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
nodejs-dev-1-build  0/1     Error      0          7h
nodejs-dev-2-build  0/1     Completed   0          6h
nodejs-dev-3-build  0/1     Completed   0          1h
nodejs-dev-3-m7wvj  1/1     Running   0          46s
nodejs-dev-4-build  0/1     Completed   0          3m
```

Después de una cuarta compilación, la tercera implementación tiene un estado de **Running** (En ejecución).

12.6. Use el comando **curl** para probar la aplicación. La aplicación muestra un mensaje **Hello World** (Hola, mundo) que contiene el nombre de host del pod de la aplicación:

```
[student@workstation nodejs-app]$ curl \
> nodejs-dev-${RHT_OCP4_DEV_USER}-nodejs-app.${RHT_OCP4_WILDCARD_DOMAIN}
Hello World from pod: nodejs-dev-3-m7wvj
```

Evaluación

Evalúe su trabajo ejecutando el comando **lab troubleshoot-review grade** de su máquina **workstation**. Corrija los errores informados y vuelva a ejecutar el script hasta obtener un resultado satisfactorio.

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

Finalizar

En **workstation**, ejecute el comando **lab troubleshoot-review finish** para terminar este trabajo de laboratorio.

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Aplicaciones, normalmente actividad de registro, como eventos, advertencias y errores, para ayudar al análisis del comportamiento de la aplicación.
- Las aplicaciones del contenedor deben imprimir los datos de registro en la salida estándar, en lugar de en un archivo, para permitir un fácil acceso a los registros.
- Para revisar los registros de un contenedor implementado localmente con Podman, use el comando **podman logs**.
- Use el comando **oc logs** para acceder a los registros para los objetos **BuildConfig** y **DeploymentConfig**, así como pods individuales dentro de un proyecto OpenShift.
- La opción **-f** le permite monitorear la salida del registro casi en tiempo real tanto para los comandos **podman logs** y **oc logs**.
- Use el comando **oc port-forward** para conectarse directamente a un puerto en un pod de aplicación. Solo debe aprovechar esta técnica en pods que no sean de producción, porque las interacciones pueden alterar el comportamiento del pod.

capítulo 9

Revisión exhaustiva

Meta

Revisar tareas de *Red Hat OpenShift I: Containers & Kubernetes*.

Objetivos

- Revisar tareas de *Red Hat OpenShift I: Containers & Kubernetes*.

Secciones

- Revisión completa

Trabajo de laboratorio

- Revisión completa de *Introduction to Containers, Kubernetes, and Red Hat OpenShift*

Revisión completa

Objetivos

Tras finalizar esta sección, debe ser capaz de demostrar las habilidades y los conocimientos aprendidos en *Red Hat OpenShift I: Containers & Kubernetes*.

Revisión de Red Hat OpenShift I: Containers & Kubernetes

Antes de comenzar el trabajo de laboratorio de revisión integral para este curso, los estudiantes deberían sentirse cómodos con los temas que se explican en los siguientes capítulos.

Capítulo 1, Introducción a la tecnología de contenedores

Describir el modo en que las aplicaciones pueden ejecutarse en contenedores organizados por Red Hat OpenShift Container Platform.

- Describir la diferencia entre las aplicaciones de contenedor y las implementaciones tradicionales.
- Describir los aspectos básicos de la arquitectura de contenedores.
- Describir los beneficios de organizar las aplicaciones y de OpenShift Container Platform.

Capítulo 2, Creación de servicios contenerizados

Aprovisionar un servicio con la tecnología de contenedores.

- Crear un servidor de base de datos a partir de una imagen de contenedor.

Capítulo 3, Administración de contenedores

Modificar las imágenes de contenedores creadas previamente para crear y administrar servicios en contenedores.

- Administrar el ciclo de vida de un contenedor desde su creación hasta su eliminación.
- Guardar los datos de la aplicación de contenedor con almacenamiento persistente.
- Describir cómo usar el reenvío de puertos para acceder a un contenedor.

Capítulo 4, Administración de imágenes de contenedores

Administrar el ciclo de vida de la imagen de un contenedor desde su creación hasta su eliminación.

- Buscar imágenes en registros remotos y extraerlas.
- Exportar, importar y administrar imágenes de contenedores de forma local y en un registro.

Capítulo 5, Creación de imágenes de contenedores personalizadas

Diseñar y codificar un Dockerfile para compilar una imagen de contenedor personalizada.

- Describir los enfoques para crear imágenes de contenedores personalizadas.
- Crear una imagen de contenedor con comandos Dockerfile comunes.

Capítulo 6, Implementación de aplicaciones contenedorizadas en OpenShift

Implementar aplicaciones con un solo contenedor en OpenShift Container Platform.

- Describir la arquitectura de Kubernetes y Red Hat OpenShift Container Platform.
- Crear recursos de Kubernetes estándares.
- Crear una ruta a un servicio.
- Compilar una aplicación con la utilidad de fuente a imagen (source-to-image) de OpenShift Container Platform.
- Crear una aplicación con la consola web de OpenShift.

Capítulo 7, Implementación de aplicaciones con múltiples contenedores

Implementar aplicaciones que estén en contenedores con varias imágenes de contenedores.

- Describir las consideraciones para organizar las aplicaciones en contenedores con varias imágenes de contenedores.
- Implementar una aplicación de varios contenedores en OpenShift usando una plantilla.

Capítulo 8, Solución de problemas de aplicaciones en contenedores

Solucionar los problemas de una aplicación en contenedores implementada en OpenShift.

- Solucionar problemas de implementación y compilación de aplicaciones en OpenShift.
- Implementar técnicas para solucionar problemas y depurar las aplicaciones en contenedores.

Sugerencias generales sobre contenedores, Kubernetes y OpenShift

Estas sugerencias pueden ahorrar tiempo en la realización del trabajo de laboratorio de revisión integral:

- El comando **podman** le permite compilar, ejecutar y administrar imágenes de contenedores. Use el comando **man podman** para acceder a la documentación de Podman. Use el comando **man podman subcommand** para obtener más información sobre cada subcomando.
- El comando **oc** le permite crear y administrar recursos de OpenShift. Use los comandos **man oc** o **oc help** para acceder a documentación de líneas de comandos de OpenShift. A continuación, se detallan algunos de los comandos de OpenShift que son particularmente útiles:

oc login -u <username> -p <password> <master_api_url>

Inicie sesión en OpenShift con el usuario especificado. Encuentre las credenciales y la URI de la API maestra en la página de laboratorio.

oc new-project

Cree un nuevo proyecto (*espacio de nombres*) para que contenga recursos de OpenShift.

oc project

Seleccione el proyecto actual (*espacio de nombres*) al cual se aplican todos los comandos posteriores.

oc create -f

Cree un recurso desde un archivo.

oc process

Procese un archivo de plantilla aplicando los valores de los parámetros a cada recurso incluido. Cree esos recursos con el comando **oc create**.

oc get

Muestre el estado y los atributos de tiempo de ejecución de los recursos de OpenShift.

oc describe

Muestre información detallada acerca de recursos de OpenShift.

oc delete

Elimine recursos de OpenShift.

- Antes de montar los volúmenes en el host de Podman y OpenShift, asegúrese de aplicar el contexto de SELinux correcto al directorio. El contexto correcto es **container_file_t**. Además, asegúrese de que se establezcan la titularidad y los permisos del directorio de acuerdo con la directiva **USER** en el Dockerfile que se usó para compilar el contenedor que se está implementando. La mayor parte del tiempo, tendrá que usar el UID y el GID numéricos en lugar del nombre de usuario y nombre de grupo para ajustar la titularidad y los permisos del directorio de volúmenes.
- En esta aula, todos los repositorios RPM se definen localmente. Debe configurar las definiciones del repositorio en una imagen de contenedor personalizada (Dockerfile) antes de ejecutar comandos **yum**.
- Cuando ejecute comandos en un Dockerfile, combine tantos comandos relacionados como sea posible en una directiva **RUN**. Esto reduce la cantidad de capas de imágenes en la imagen de contenedor.
- Una práctica recomendada para el diseño de un Dockerfile incluye el uso de variables de entorno para especificar constantes repetidas en el archivo.

► Trabajo de laboratorio

Organizar en contenedores e implementar una aplicación de software

En esta revisión, contenerizará un servidor Nexus, lo compilará y probará con Podman, y lo implementará en un clúster de OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Escribir un Dockerfile que organice un servidor Nexus en contenedores de manera correcta.
- Compilar una imagen de contenedor del servidor Nexus e implementarla con Podman.
- Implementar la imagen de contenedor del servidor Nexus en un clúster de OpenShift.

Antes De Comenzar

Ejecutar un script de configuración para esta revisión completa.

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

Los archivos del trabajo de laboratorio están ubicados en el directorio **/home/student/D0180/labs/comprehensive-review**. Los archivos de la solución están ubicados en el directorio **/home/student/D0180/solutions/comprehensive-review**.

Instrucciones

Use los siguientes pasos para crear y probar un servidor Nexus en contenedores tanto localmente como en OpenShift:

Pasos

1. Cree una imagen de contenedor que inicie una instancia de un servidor Nexus:
 - El directorio **/home/student/D0180/labs/comprehensive-review/image** contiene archivos para compilar la imagen del contenedor. Ejecute el script **get-nexus-bundle.sh** para recuperar los archivos del servidor Nexus.
 - Escriba un Dockerfile que organice el servidor Nexus en contenedores. El Dockerfile debe estar ubicado en el directorio **/home/student/D0180/labs/comprehensive-review/image**. El Dockerfile también debe:
 - Usar una imagen base de **ubi7/ubi:7.7** y establecer un mantenedor arbitrario.
 - Establecer la variable de entorno **NEXUS_VERSION** en **2.14.3-02** y establecer **NEXUS_HOME** en **/opt/nexus**.
 - Instalar el paquete **java-1.8.0-openjdk-devel**.

Los repositorios de RPM se configuran en el archivo **training.repo** provisto. Asegúrese de agregar este archivo al contenedor en el directorio **/etc/yum.repos.d**.

- Ejecute un comando para crear un usuario y un grupo **nexus**. Tienen un UID y un GID de **1001**.

- Desempaque el archivo **nexus-2.14.3-02-bundle.tar.gz** en el directorio **`\${NEXUS_HOME}`/**. Agregue **nexus-start.sh** al mismo directorio.

Ejecute un comando, **ln -s \${NEXUS_HOME}/nexus-\${NEXUS_VERSION} \${NEXUS_HOME}/nexus2**, para crear un enlace simbólico en el contenedor. Ejecute un comando para cambiar de manera recursiva la propiedad del directorio principal de Nexus a **nexus:nexus**.

- Haga que el contenedor se ejecute con el usuario **nexus** y establezca el directorio de trabajo **/opt/nexus**.
- Defina un punto de montaje de volumen para el directorio contenedor **/opt/nexus/sonatype-work**. El servidor Nexus almacena datos en este directorio.
- Establezca el comando contenedor predeterminado en **nexus-start.sh**.

Hay dos archivos ***.snippet** en el directorio **/home/student/D0180/labs/comprehensive-review/images** que proporcionan los comandos necesarios para crear la cuenta de Nexus e instalar Java. Use los archivos como ayuda para escribir el Dockerfile.

- Compile la imagen del contenedor con el nombre **nexus**.
2. Compile y pruebe la imagen de contenedor mediante Podman con un montaje de volumen.
 - Use el script **/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh** para iniciar un nuevo contenedor con un volumen de montaje.
 - Revise los registros del contenedor para verificar que el servidor esté iniciado y en ejecución.
 - Pruebe el acceso al servicio de contenedor usando la URL: **http://<container IP address>:8081/nexus**.
 - Elimine el contenedor de prueba.
 3. Implemente la imagen de contenedor del servidor Nexus en un clúster de OpenShift. Debe hacer lo siguiente:
 - Etiquetar la imagen del contenedor del servidor Nexus como **quay.io/\${RHT_OCP4_QUAY_USER}/nexus:latest** y enviarla al registro privado.
 - Crear un proyecto de OpenShift con el nombre **`\${RHT_OCP4_DEV_USER}-review**.
 - Procesar la plantilla **deploy/openshift/resources/nexus-template.json** y crear los recursos de Kubernetes.
 - Crear una ruta para el servicio Nexus. Verifique que puede acceder a **http://nexus-`\${RHT_OCP4_DEV_USER}-review.\${RHT_OCP4_WILDCARD_DOMAIN}/nexus/** desde **workstation**.

Evaluación

Después de implementar la imagen de contenedor del servidor Nexus en el clúster de OpenShift, verifique su trabajo mediante la ejecución del script de evaluación del trabajo de laboratorio:

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

Finalizar

En **workstation**, ejecute el comando **lab comprehensive-review finish** para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

► Solución

Organizar en contenedores e implementar una aplicación de software

En esta revisión, contenerizará un servidor Nexus, lo compilará y probará con Podman, y lo implementará en un clúster de OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Escribir un Dockerfile que organice un servidor Nexus en contenedores de manera correcta.
- Compilar una imagen de contenedor del servidor Nexus e implementarla con Podman.
- Implementar la imagen de contenedor del servidor Nexus en un clúster de OpenShift.

Andes De Comenzar

Ejecutar un script de configuración para esta revisión completa.

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

Los archivos del trabajo de laboratorio están ubicados en el directorio **/home/student/D0180/labs/comprehensive-review**. Los archivos de la solución están ubicados en el directorio **/home/student/D0180/solutions/comprehensive-review**.

Instrucciones

Use los siguientes pasos para crear y probar un servidor Nexus en contenedores tanto localmente como en OpenShift.

Pasos

1. Cree una imagen de contenedor que inicie una instancia de un servidor Nexus:
 - El directorio **/home/student/D0180/labs/comprehensive-review/image** contiene archivos para compilar la imagen del contenedor. Ejecute el script **get-nexus-bundle.sh** para recuperar los archivos del servidor Nexus.
 - Escriba un Dockerfile que organice el servidor Nexus en contenedores. El Dockerfile debe estar ubicado en el directorio **/home/student/D0180/labs/comprehensive-review/image**. El Dockerfile también debe:
 - Usar una imagen base de **ubi7/ubi:7.7** y establecer un mantenedor arbitrario.
 - Establecer la variable de entorno **NEXUS_VERSION** en **2.14.3-02** y establecer **NEXUS_HOME** en **/opt/nexus**.
 - Instalar el paquete **java-1.8.0-openjdk-devel**.

Los repositorios de RPM se configuran en el archivo **training.repo** provisto. Asegúrese de agregar este archivo al contenedor en el directorio **/etc/yum.repos.d**.

- Ejecute un comando para crear un usuario y un grupo **nexus**. Tienen un UID y un GID de **1001**.

- Desempaque el archivo **nexus-2.14.3-02-bundle.tar.gz** en el directorio **\${NEXUS_HOME} /**. Agregue **nexus-start.sh** al mismo directorio.

Ejecute un comando, **ln -s \${NEXUS_HOME}/nexus-\${NEXUS_VERSION} \${NEXUS_HOME}/nexus2**, para crear un enlace simbólico en el contenedor. Ejecute un comando para cambiar de manera recursiva la propiedad del directorio principal de Nexus a **nexus:nexus**.

- Haga que el contenedor se ejecute con el usuario **nexus** y establezca el directorio de trabajo **/opt/nexus**.
- Defina un punto de montaje de volumen para el directorio contenedor **/opt/nexus/sonatype-work**. El servidor Nexus almacena datos en este directorio.
- Establezca el comando contenedor predeterminado en **nexus-start.sh**.

Hay dos archivos ***.snippet** en el directorio **/home/student/D0180/labs/comprehensive-review/images** que proporcionan los comandos necesarios para crear la cuenta de Nexus e instalar Java. Use los archivos como ayuda para escribir el Dockerfile.

- Compile la imagen del contenedor con el nombre **nexus**.
- 1.1. Ejecute el script **get-nexus-bundle.sh** para recuperar los archivos del servidor 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. Escriba un Dockerfile que organice el servidor Nexus en contenedores. Vaya al directorio **/home/student/D0180/labs/comprehensive-review/image** y cree el Dockerfile.

- 1.2.1. Especifique la imagen base que se usará:

```
FROM ubi7/ubi:7.7
```

- 1.2.2. Escriba un nombre y correo electrónico arbitrarios como mantenedor:

```
FROM ubi7/ubi:7.7  
MAINTAINER username <username@example.com>
```

- 1.2.3. Configure las variables de entorno para **NEXUS_VERSION** y **NEXUS_HOME**:

```
FROM ubi7/ubi:7.7
MAINTAINER username <username@example.com>

ENV NEXUS_VERSION=2.14.3-02 \
    NEXUS_HOME=/opt/nexus
```

- 1.2.4. Agregue el repositorio **training.repo** al directorio **/etc/yum.repos.d**. Instale el paquete Java mediante el comando **yum**.

```
...
ENV NEXUS_VERSION=2.14.3-02 \
    NEXUS_HOME=/opt/nexus

RUN yum install -y --setopt=tsflags=nodocs java-1.8.0-openjdk-devel && \
    yum clean all -y
```

- 1.2.5. Cree el directorio principal del servidor, así como la cuenta y el grupo de servicio. Haga que el directorio de inicio sea propiedad de la cuenta de servicio.

```
...
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}
```

- 1.2.6. Haga que el contenedor se ejecute como el usuario **nexus**.

```
...
USER nexus
```

- 1.2.7. Instale el software del servidor Nexus en **NEXUS_HOME** y agregue el script de arranque. Observe que la directiva **ADD** extraerá los archivos Nexus. Cree el enlace simbólico **nexus2** que apunta al directorio del servidor Nexus. Cambiar de manera recursiva la propiedad del directorio **\${NEXUS_HOME}** a **nexus: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
```

- 1.2.8. Haga que **/opt/nexus** sea el directorio de trabajo actual:

```
...
WORKDIR ${NEXUS_HOME}
```

1.2.9. Defina un punto de montaje del volumen para almacenar los datos persistentes del servidor Nexus:

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

1.2.10. Establezca la instrucción **CMD** en **["sh", "nexus-start.sh"]**. A continuación, se muestra el contenido del Dockerfile completo:

```
FROM ubi7/ubi:7.7

MAINTAINER username <username@example.com>

ENV NEXUS_VERSION=2.14.3-02 \
    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. Compile la imagen del contenedor con el nombre **nexus**.

```
[student@workstation image]$ sudo podman build -t nexus .
STEP 1: FROM ubi7/ubi:7.7
Getting image source signatures
...output omitted...
STEP 25: COMMIT nexus
```

2. Compile y pruebe la imagen de contenedor mediante Podman con un montaje de volumen.

- Use el script **/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh** para iniciar un nuevo contenedor con un volumen de montaje.

capítulo 9 | Revisión exhaustiva

- Revise los registros del contenedor para verificar que el servidor esté iniciado y en ejecución.
 - Pruebe el acceso al servicio de contenedor usando la URL: `http://<container IP address>:8081/nexus`.
 - Elimine el contenedor de prueba.
- 2.1. Ejecute el script **run-persistent.sh**. Reemplace el nombre del contenedor como se muestra en la salida del comando **podman ps**.

```
[student@workstation images]$ cd /home/student/DO180/labs/comprehensive-review  
[student@workstation comprehensive-review]$ cd deploy/local  
[student@workstation local]$ ./run-persistent.sh  
80970007036bbb313d8eeb7621fada0ed3f0b4115529dc50da4dccef0da34533
```

- 2.2. Revise los registros del contenedor para verificar que el servidor esté iniciado y en ejecución.

```
[student@workstation local]$ sudo podman ps \  
> --format="table {{.ID}} {{.Names}} {{.Image}}"  
CONTAINER ID NAMES IMAGE  
81f480f21d47 inspiring_poincare localhost/nexus:latest  
[student@workstation local]$ sudo podman logs -f inspiring_poincare  
...output omitted...  
... INFO [jetty-main-1] ...jetty.JettyServer - Running  
... INFO [main] ...jetty.JettyServer - Started  
Ctrl+C
```

- 2.3. Inspeccione el contenedor en ejecución para determinar su dirección IP. Proporcione esta dirección IP al comando **curl** para probar el contenedor.

```
[student@workstation local]$ sudo podman inspect \  
> -f '{{.NetworkSettings.IPAddress}}' inspiring_poincare  
10.88.0.12  
[student@workstation local]$ curl -v 10.88.0.12:8081/nexus/  
About to connect() to 10.88.0.12 port 8081 (#0)  
* Trying 10.88.0.12...  
* Connected to 10.88.0.12 (10.88.0.12) port 8081 (#0)  
> GET /nexus/ HTTP/1.1  
> User-Agent: curl/7.29.0  
> Host: 10.88.0.12:8081  
> Accept: */*  
>  
< HTTP/1.1 200 OK  
< Date: Tue, 05 Mar 2019 16:59:30 GMT  
< Server: Nexus/2.14.3-02  
...output omitted...  
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">  
<head>  
<title>Nexus Repository Manager</title>  
...output omitted...
```

- 2.4. Elimine el contenedor de prueba.

```
[student@workstation local]$ sudo podman kill inspiring_poincare  
81f480f21d475af683b4b003ca6e002d37e6aaa581393d3f2f95a1a7b7eb768b
```

3. Implemente la imagen de contenedor del servidor Nexus en un clúster de OpenShift. Debe hacer lo siguiente:
- Etiquetar la imagen del contenedor del servidor Nexus como **quay.io/\${RHT_OCP4_QUAY_USER}/nexus:latest** y enviarla al registro privado.
 - Crear un proyecto de OpenShift con el nombre **\${RHT_OCP4_DEV_USER}-review**.
 - Procesar la plantilla **deploy/openshift/resources/nexus-template.json** y crear los recursos de Kubernetes.
 - Crear una ruta para el servicio Nexus. Verifique que puede acceder a **http://nexus-\${RHT_OCP4_DEV_USER}-review.\${RHT_OCP4_WILDCARD_DOMAIN}/nexus/** desde **workstation**.

- 3.1. Inicie sesión en su cuenta de Quay.io.

```
[student@workstation local]$ sudo podman login -u ${RHT_OCP4_QUAY_USER} quay.io  
Password: your_quay_password  
Login Succeeded!
```

- 3.2. Publique la imagen de contenedor del servidor Nexus en su registro de quay.io:

```
[student@workstation local]$ sudo 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. De forma predeterminada, los repositorios creados mediante el envío de imágenes a quay.io son privados. Consulte la sección de **Repositories Visibility** (Visibilidad de repositorios) del Apéndice C para conocer los detalles acerca de cómo cambiar la visibilidad del repositorio.

- 3.4. Cree el proyecto de OpenShift:

```
[student@workstation local]$ cd ~/D0180/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. Procese la plantilla y cree los recursos de Kubernetes:

```
[student@workstation openshift]$ oc process -f resources/nexus-template.json \
> -p RHT_OCP4_QUAY_USER=${RHT_OCP4_QUAY_USER} \
> | oc create -f -
service/nexus created
persistentvolumeclaim/nexus created
deploymentconfig.apps.openshift.io/nexus created
[student@workstation openshift]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
nexus-1-wk8rv  1/1     Running   1          1m
nexus-1-deploy 0/1     Completed  0          2m
```

3.6. Exponga el servicio con la creación de una ruta:

```
[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-your_dev_username-review.your_wildcard_domain
...output omitted...
```

3.7. Use un navegador para conectarse con la aplicación web del servidor Nexus en `http://nexus-$\{RHT_OCP4_DEV_USER\}-review.\$\{RHT_OCP4_WILDCARD_DOMAIN\}/nexus/`.

Evaluación

Después de implementar la imagen de contenedor del servidor Nexus en el clúster de OpenShift, verifique su trabajo mediante la ejecución del script de evaluación del trabajo de laboratorio:

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

Finalizar

En `workstation`, ejecute el comando `lab comprehensive-review finish` para terminar este trabajo de laboratorio.

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

Esto concluye el trabajo de laboratorio.

apéndice A

Implementación de la arquitectura de microservicios

Meta

Refactorizar una aplicación en microservicios.

Objetivos

- Dividir una aplicación entre varios contenedores para separar distintas capas y servicios.

Secciones

- Implementación de arquitecturas de microservicios (con ejercicio guiado)

Implementación de las arquitecturas de microservicios

Objetivos

Tras finalizar esta sección, usted deberá ser capaz de realizar lo siguiente:

- Dividir una aplicación entre varios contenedores para separar distintas capas y servicios.
- Describir enfoques típicos para descomponer una aplicación monolítica en varias unidades implementables.
- Describir el modo de descomponer la aplicación To Do List (Lista de tareas) en tres contenedores que coincidan con sus niveles lógicos.

Beneficios de descomponer una aplicación monolítica en contenedores

Por lo general, el desarrollo tradicional de aplicaciones tiene muchas funciones distintas empaquetadas como una sola unidad de implementación o una aplicación *monolítica*. El desarrollo tradicional también puede implementar servicios de soporte, como bases de datos y otros servicios de middleware, en el mismo servidor que la aplicación. Si bien las aplicaciones monolíticas aún se pueden implementar en un contenedor, muchas de las ventajas de una arquitectura de contenedores, como la escalabilidad y la agilidad, no son tan frecuentes. Descomponer los monolitos requiere extremo cuidado, y se recomienda que, en las aplicaciones de microservicios, cada microservicio ejecute la funcionalidad mínima que se puede ejecutar de forma aislada en cada contenedor.

Tener contenedores más pequeños y descomponer una aplicación y sus servicios de soporte en varios contenedores proporcionan muchas ventajas, como las siguientes:

- Mayor utilización del hardware, ya que los contenedores más pequeños son más fáciles de adaptarse a la capacidad del host disponible.
- Escalamiento (scaling) más sencillo, ya que se pueden escalar partes de la aplicación para soportar una mayor carga de trabajo sin escalar otras partes de la aplicación.
- Actualizaciones más sencillas, ya que los desarrolladores pueden actualizar partes de la aplicación sin afectar a otras partes de la misma aplicación.

Las siguientes son dos formas populares de descomponer una aplicación:

- Niveles: sobre la base de capas de arquitectura.
- Servicios: sobre la base de la funcionalidad de la aplicación.

División sobre la base de capas (niveles)

Una manera común en que los desarrolladores organizan las aplicaciones es en niveles, según cuán cerca de los usuarios finales y cuán lejos de los almacenes de datos están las funciones. Un buen ejemplo de la arquitectura tradicional de tres niveles es la presentación, la lógica empresarial y la persistencia.

Esta arquitectura lógica generalmente corresponde a una arquitectura de implementación física, donde la capa de presentación se implementaría en un servidor web; la capa empresarial, en un servidor de aplicaciones; y la capa de persistencia, en un servidor de base de datos.

Descomponer una aplicación sobre la base de niveles permite a los desarrolladores especializarse en tecnologías específicas basadas en los niveles de la aplicación. Por ejemplo, algunos desarrolladores se centran en aplicaciones web, mientras que otros prefieren el desarrollo de bases de datos. Otra ventaja es la capacidad de proporcionar implementaciones por niveles alternativas sobre la base de diferentes tecnologías; por ejemplo, la creación de una aplicación móvil como otro front-end para una aplicación existente. La aplicación móvil sería un nivel de presentación alternativo, que reutiliza el nivel empresarial y el nivel de persistencia de la aplicación web original.

Generalmente, las aplicaciones más pequeñas tienen el nivel empresarial y el nivel de presentación implementados como una sola unidad. Por ejemplo, en el mismo servidor web, pero a medida que aumenta la carga, la capa de presentación se mueve a su propia unidad de implementación para distribuir la carga. Las aplicaciones más pequeñas incluso pueden integrarse en la base de datos. Los desarrolladores a menudo compilan e implementan las aplicaciones más demandantes de esta manera monolítica.

Cuando los desarrolladores descomponen una aplicación monolítica en niveles, usualmente aplican varios cambios:

- Los parámetros de conexión con una base de datos y otros servicios de middleware, como mensajería, estaban codificados con nombres de hosts o direcciones IP fijas, en general, **localhost**. Deben parametrizarse para apuntar a servidores externos que pueden ser diferentes desde desarrollo hasta producción.
- En el caso de las aplicaciones web, las llamadas de Ajax no pueden realizarse con URL relativas. Deben usar una URL absoluta que apunte a un nombre de host de DNS público fijo.
- Los navegadores web modernos rechazan las llamadas de Ajax a servidores que sean diferentes al que contiene el script que realiza la llamada, como medida de seguridad. La aplicación debe tener permisos para el *Uso compartido de recursos entre orígenes* (CORS).

Después de que se dividan los niveles de la aplicación para poder ejecutarse desde diferentes servidores, no debería haber problemas al ejecutarlos desde diferentes contenedores.

División sobre la base de servicios discretos

Las aplicaciones más complejas están compuestas por muchos servicios semiindependientes. Por ejemplo, un almacén en línea tendría un catálogo de productos, un carrito de compras, pagos, envíos, etc.

Cuando un servicio específico en una aplicación monolítica se degrada, escalar el servicio para mejorar el rendimiento implica escalar todos los demás servicios constitutivos de la aplicación. Sin embargo, si el servicio degradado forma parte de una arquitectura de microservicios, el servicio afectado se escala independientemente de los otros servicios de la aplicación. En la siguiente figura, se ilustra el escalamiento de servicio tanto para una arquitectura monolítica como para una arquitectura basada en microservicios:

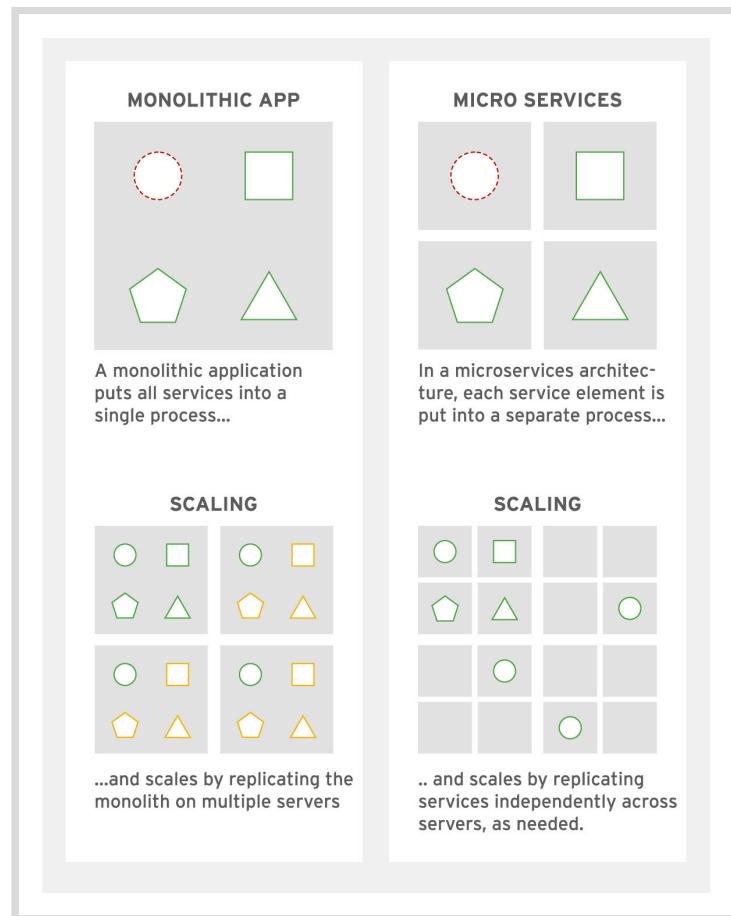


Figura A.1: Comparación del escalamiento de aplicaciones en una arquitectura monolítica versus una arquitectura de microservicios

Tanto las *arquitecturas orientadas a servicios* (SOA) como las arquitecturas de *microservicios* más recientes empaquetan e implementan estos conjuntos de funciones como unidades diferentes. Esto permite que cada conjunto de funciones sea desarrollado por su propio equipo, se actualice y se escale sin perjudicar otros conjuntos de funciones (o servicios). Las preocupaciones entre funciones, como la autenticación, también se pueden empaquetar e implementar como servicios que consumen otras implementaciones de servicios.

La división de cada asunto en un servidor separado puede producir muchas aplicaciones. Tienen una arquitectura lógica, se empaquetan y se implementan como una pequeña cantidad de unidades, algunas veces, incluso, como una sola unidad monolítica que usa un enfoque de servicio.

Los contenedores permiten que las arquitecturas basadas en servicios se materialicen durante la implementación. Esa es la razón por la cual normalmente se unen los microservicios y contenedores. Sin embargo, los contenedores por sí solos no son suficientes; deben complementarse con herramientas de orquestación para administrar las dependencias entre servicios.

La arquitectura de los microservicios lleva al extremo las arquitecturas basadas en servicios. Un servicio tiene el tamaño más pequeño que puede tener (sin descomponer un conjunto de funciones) y se implementa y se administra como una unidad independiente, en lugar de como parte de una aplicación más grande. Esto permite reutilizar los microservicios existentes para crear nuevas aplicaciones.

Para descomponer una aplicación en servicios, necesita el mismo tipo de cambio que cuando se descompone en niveles; por ejemplo, parametrizar los parámetros de conexión en las bases de datos y otros servicios de middleware y ocuparse de las protecciones de seguridad del navegador web.

Refactorización de la aplicación To Do List (Lista de tareas)

La aplicación To Do List (Lista de tareas) es una aplicación simple con un solo conjunto de funciones, de modo que descomponerla en servicios no es verdaderamente significativo. Sin embargo, la refactorización de esta en el nivel empresarial y el nivel de presentación, es decir, en un front-end y un back-end para implementar en diferentes contenedores, ilustra el mismo tipo de cambios que necesitaría una aplicación típica para descomponerse en servicios.

En la siguiente figura, se muestra la aplicación To Do List (Lista de tareas) implementada en tres contenedores, uno para cada nivel:

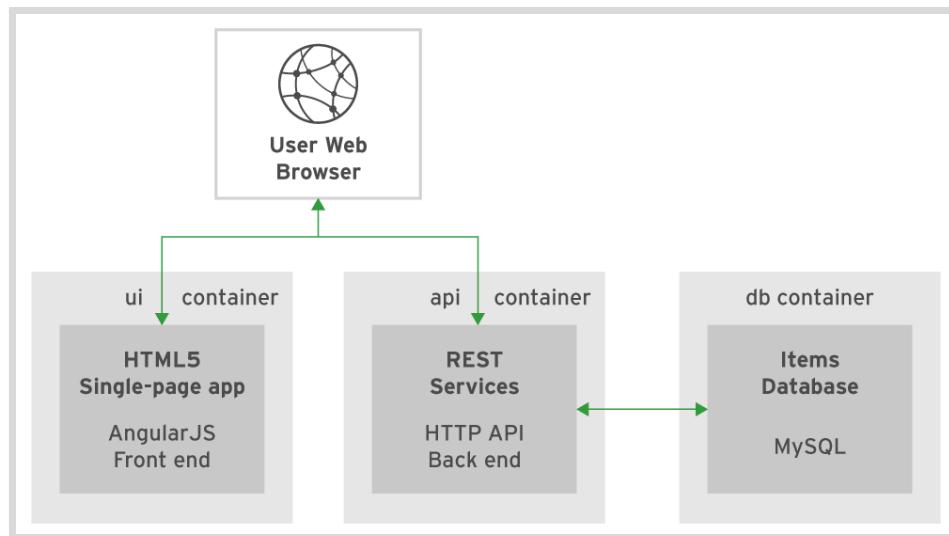


Figura A.2: La aplicación To Do List (Lista de tareas) descompuesta en niveles y cada uno implementado como contenedor

Si comparamos el código fuente de la aplicación monolítica original con el nuevo refactorizado, esta es una descripción general de los cambios:

- JavaScript de front-end en **script/items.js** usa `workstation.lab.example.com` como nombre de host para conectarse con el back-end.
- El back-end usa variables de entorno para obtener los parámetros de conexión de la base de datos.
- El back-end debe responder las solicitudes con el verbo **OPTIONS** (OPCIONES) HTTP con encabezados que indiquen al navegador web que acepte solicitudes que provengan de diferentes dominios de DNS con CORS .

Otras versiones del servicio de back-end pueden tener cambios similares. Cada lenguaje de programación y el marco (framework) de REST tienen su propia sintaxis y sus propias características.



Referencias

Página de la aplicación monolítica en Wikipedia

https://en.wikipedia.org/wiki/Monolithic_application

Página de CORS en Wikipedia

https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

► Ejercicio Guiado

Refactorización de la aplicación To Do List (Lista de tareas)

En este trabajo de laboratorio, refactorizará la aplicación To Do List (Lista de tareas) en varios contenedores que están vinculados, y permitirá que la aplicación de front-end HTML 5, la API REST Node.js y el servidor de MySQL se ejecuten en sus propios contenedores.

Resultados

Debería ser capaz de refactorizar una aplicación monolítica en sus niveles e implementar cada nivel como un microservicio.

Andes De Comenzar

Ejecute el siguiente comando para configurar los directorios de trabajo para el trabajo de laboratorio con los archivos de la aplicación To Do List (Lista de tareas).

```
[student@workstation ~]$ lab appendix-microservices start
```

► 1. Mover los archivos HTML

El primer paso para refactorizar la aplicación To Do List (Lista de tareas) es mover el código de front-end desde la aplicación a su propio contenedor en ejecución. Este paso lo guía durante el movimiento de la aplicación HTML y sus archivos dependientes a su propio directorio para la implementación en un servidor de Apache que se ejecuta en un contenedor.

- 1.1. Mueva los archivos estáticos y HTML al directorio **src/** desde la aplicación To Do List (Lista de tareas) de Node.js monolítico:

```
[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. La aplicación de front-end actual interactúa con el servicio de la API con una URL relativa. Dado que la API y el código de front-end ahora se ejecutan en contenedores separados, el front-end debe ajustarse para apuntar a la URL absoluta de la API de la aplicación To Do List (Lista de tareas).

Abra el archivo **/home/student/D0180/labs/appendix-microservices/apps/html5/src/script/item.js**. En la parte inferior del archivo, busque el siguiente método:

```
app.factory('itemService', function ($resource) {
  return $resource('api/items/:id');
});
```

Reemplace el código con el siguiente contenido:

```
app.factory('itemService', function ($resource) {
    return $resource('http://workstation.lab.example.com:30080/todo/api/
items/:id');
});
```

Asegúrese de que no haya saltos de línea en la nueva URL, guarde el archivo y salga del editor.

► 2. Compile la imagen HTML.

- 2.1. Ejecute el script de compilación para compilar la imagen principal de Apache.

```
[student@workstation html5]$ cd ~/D0180/labs/appendix-microservices/images/apache
[student@workstation apache]$ ./build.sh
STEP 1: FROM ubi7/ubi:7.7
...output omitted...
STEP 13: COMMIT do180/httpd
```

- 2.2. Verifique que la imagen se haya compilado correctamente:

```
[student@workstation apache]$ sudo podman images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
localhost/do180/httpd  latest   34376f2a318f   2 minutes ago  282.6 MB
...
```

- 2.3. Compile la imagen secundaria de Apache:

```
[student@workstation apache]$ cd ~/D0180/labs/appendix-microservices/deploy/html5
[student@workstation html5]$ ./build.sh
STEP 1: FROM do180/httpd
STEP 2: COPY ./src/ ${HOME}/
--> cf11...dde1
STEP 3: COMMIT do180/todo_frontend
```

- 2.4. Verifique que la imagen se haya compilado correctamente:

```
[student@workstation html5]$ sudo podman images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
localhost/do180/todo_frontend  latest   30b3fc531bc6   2 minutes ago  286.9 MB
localhost/do180/httpd        latest   34376f2a318f   4 minutes ago  282.6 MB
...
```

► 3. Modifique la API REST para conectarse con contenedores externos.

- 3.1. La API REST actualmente usa valores codificados para conectarse con la base de datos MySQL. Edite el archivo `/home/student/D0180/labs/appendix-microservices/apps/nodejs/models/db.js`, que contiene la configuración de la base de datos. Actualice los valores `dbname`, `username` y `password` para usar, en su lugar, variables del entorno. Además, actualice `params.host` para que apunte al nombre de host del host que ejecuta el contenedor MySQL y actualice `params.port` para reflejar el puerto redirigido al contenedor. Ambos valores están disponibles como las variables de entorno `MYSQL_SERVICE_HOST` y

MYSQL_SERVICE_PORT, respectivamente. El contenido reemplazado debe verse de la siguiente manera:

```
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'
  }
};
```



nota

Este archivo puede copiarse y pegarse desde **/home/student/D0180/solutions/appendix-microservices/apps/nodejs/models/db.js**.

- 3.2. Configure el back-end para administrar el *Uso compartido de recursos entre orígenes* (CORS). Esto sucede cuando se realiza una solicitud de recursos desde un dominio diferente a aquél desde el cual se realizó la solicitud. Dado que la API debe gestionar solicitudes desde un dominio de DNS diferente (la aplicación de front-end), es necesario crear excepciones de seguridad para permitir que estas solicitudes se realicen correctamente. Realice las siguientes modificaciones a la aplicación en el lenguaje de su preferencia para gestionar el CORS.

Agregue "**restify-cors-middleware": "1.1.1**" como nueva dependencia al archivo **package.json** ubicado en **/home/student/D0180/labs/appendix-microservices/apps/nodejs/package.json**. Recuerde colocar una coma al final de la dependencia anterior. Asegúrese de que el final del archivo se vea de la siguiente manera:

```
"sequelize": "5.21.1",
"mysql2": "2.0.0",
"restify-cors-middleware": "1.1.1"
}
```

Actualice el archivo **app.js** ubicado en **/home/student/D0180/labs/appendix-microservices/apps/nodejs/app.js** para configurar el uso de CORS. Solicite el módulo **restify-cors-middleware** en la segunda línea y, luego, actualice el contenido del archivo para que coincida con lo siguiente:

```
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());
```

```
const cors = corsMiddleware({
  origins: ['*']
});

server.pre(cors.preflight);
server.use(cors.actual);

controller.context(server, '/todo/api', model);
```

Los orígenes con el valor `["*"]` indican al servidor que permita cualquier dominio. En un servidor de producción, este valor generalmente sería un arreglo de dominios que requiere acceso a la API.

► 4. Compile la imagen de la API REST.

- 4.1. Compile la imagen hijo de la API REST con el siguiente comando. Esta imagen usa la imagen Node.js.

```
[student@workstation html5]$ cd ~/DO180/labs/appendix-microservices/deploy/nodejs
[student@workstation nodejs]$ ./build.sh
STEP 1: FROM rhscl/nodejs-4-rhel7:latest
...output omitted...
STEP 11: COMMIT do180/todonodejs
```

- 4.2. Ejecute el comando **podman images** para verificar que todas las imágenes requeridas se hayan creado correctamente:

```
[student@workstation nodejs]$ sudo podman images
REPOSITORY                  TAG      IMAGE ID      CREATED       SIZE
localhost/do180/httpd        latest   2963ca81ac51  5 seconds ago  249 MB
localhost/do180/todonodejs   latest   7b64ef105c50  7 minutes ago  533 MB
localhost/do180/todo_frontend latest   53ad57d2306c  9 minutes ago  254 MB
...output omitted...
```

► 5. Ejecute los contenedores.

- 5.1. Use el script **run.sh** para ejecutar los contenedores:

```
[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. Ejecute el comando **podman ps** para confirmar que se estén ejecutando los tres contenedores:

```
[student@workstation linked]$ sudo podman ps
... IMAGE                      ... PORTS              NAMES
... localhost/do180/todo_frontend ... 0.0.0.0:30000->80/tcp    todo_frontend
... localhost/do180/todonodejs    ... 8080/tcp, 0.0.0.0:30080... todoapi
... localhost/rhscl/mysql-57-rhel7 ... 0.0.0.0:30306->3306/tcp    mysql
```

► **6. Pruebe la aplicación.**

- 6.1. Use el comando **curl** para verificar que la API REST para la aplicación To Do List (Lista de tareas) funcione correctamente:

```
[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. Abra Firefox en **workstation** y navegue hasta <http://127.0.0.1:30000>, donde debería ver la aplicación To Do List (Lista de tareas).

Finalizar

En **workstation**, ejecute el script **lab appendix-microservices finish** para terminar este trabajo de laboratorio.

```
[student@workstation ~]$ lab appendix-microservices finish
```

Esto concluye el ejercicio guiado.

Resumen

En este capítulo, aprendió lo siguiente:

- Descomponer una aplicación monolítica en varios contenedores permite una mayor escalabilidad de la aplicación, facilita las actualizaciones y permite una mayor utilización del hardware.
- Los tres niveles comunes para la división lógica de una aplicación son el nivel de presentación, el nivel empresarial y el nivel de persistencia.
- El *Uso compartido de recursos entre orígenes* (CORS) puede impedir las llamadas de Ajax a servidores diferentes al que se usó para descargar las páginas. Asegúrese de realizar aprovisionamientos para permitir el CORS de otros contenedores en la aplicación.
- Las imágenes de contenedores están diseñadas para ser inmutables, pero las configuraciones pueden pasarse en el momento de la compilación de la imagen o mediante la creación de un almacenamiento persistente para las configuraciones.

apéndice B

Creación de una cuenta GitHub

Meta

Describir cómo crear una cuenta de GitHub para los trabajos de laboratorio del curso.

Creación de una cuenta GitHub

Objetivos

Tras finalizar esta sección, deberá ser capaz de crear una cuenta de GitHub y crear repositorios Git públicos para los trabajos de laboratorio del curso.

Creación de una cuenta GitHub

Necesita una cuenta de GitHub para crear uno o más repositorios Git *públicos* para los trabajos de laboratorio de este curso. Si ya tiene una cuenta de GitHub, puede omitir los pasos que se detallan en este apéndice.



Importante

Si ya tiene una cuenta de GitHub, asegúrese de crear solo repositorios Git *públicos* para los trabajos de laboratorio de este curso. Las instrucciones y los scripts de evaluación del trabajo de laboratorio requieren acceso no autenticado para clonar el repositorio. Los repositorios deben ser accesibles sin proporcionar contraseñas, claves SSH o claves GPG.

Para crear una nueva cuenta de GitHub, realice los siguientes pasos:

1. Diríjase a <https://github.com> mediante un navegador web.
2. Ingrese los detalles necesarios y, a continuación, haga clic en **Sign up for GitHub** (Registrarse en GitHub).

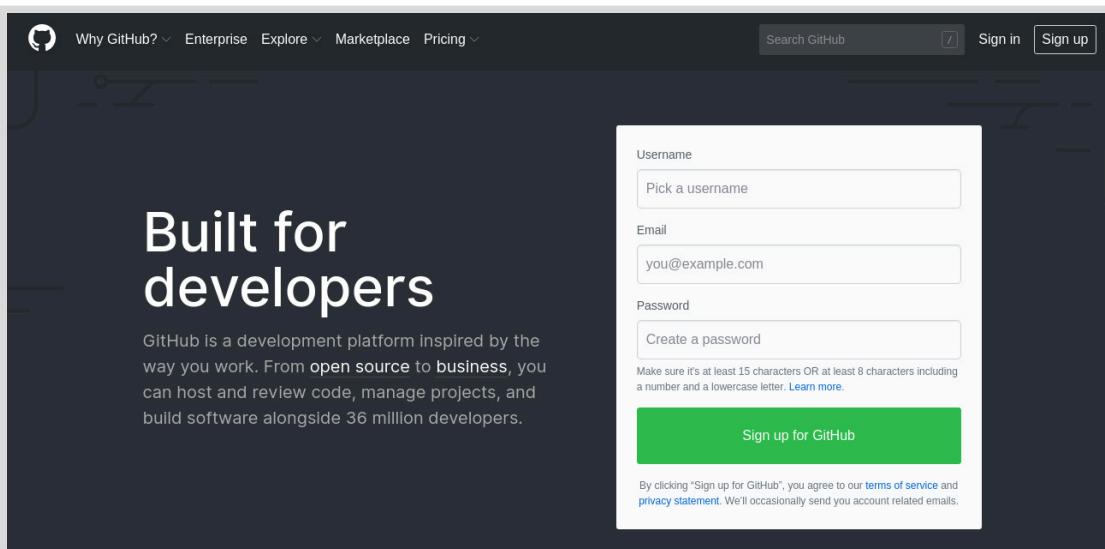


Figura B.1: Creación de una cuenta de GitHub

3. Recibirá un correo electrónico con instrucciones sobre cómo activar su cuenta de GitHub. Verifique su dirección de correo electrónico y, luego, inicie sesión en el sitio web de GitHub con el nombre de usuario y la contraseña que proporcionó durante la creación de la cuenta.

4. Despues de iniciar sesión en GitHub, puede crear nuevos repositorios Git con un clic en **New** (Nuevo) en el panel **Repositories** (Repositorios) a la izquierda de la página principal de GitHub.

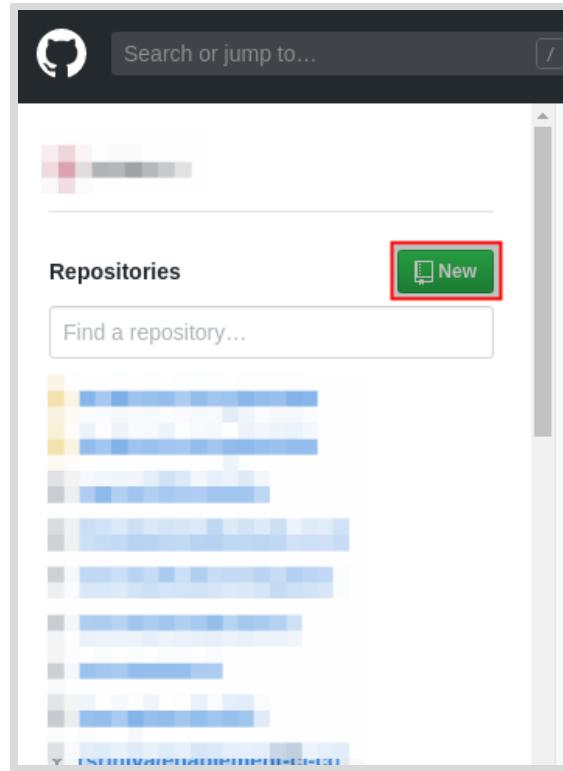


Figura B.2: Creación de un nuevo repositorio Git

De manera alternativa, haga clic en el ícono con el signo más (+) en la esquina superior derecha (a la derecha del ícono de campana) y, luego, haga clic en **New Repository (Nuevo repositorio)**.

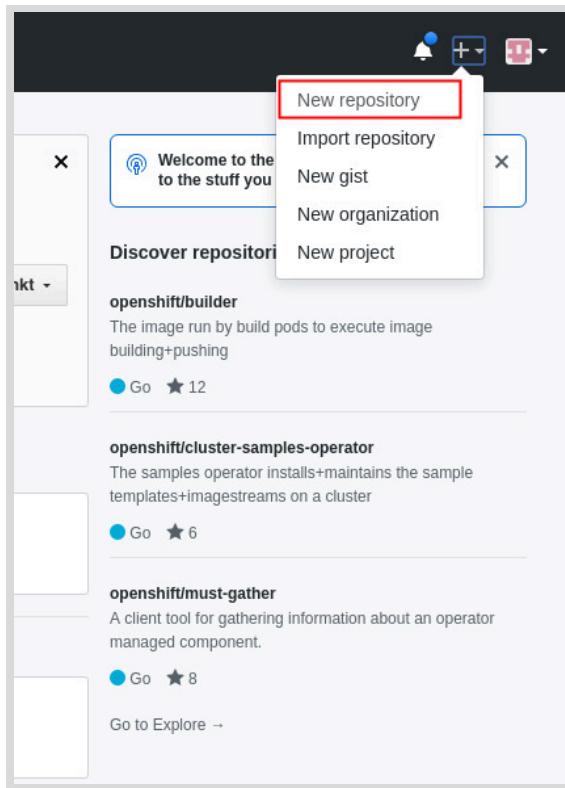


Figura B.3: Creación de un nuevo repositorio Git



Referencias

Registro para obtener una nueva cuenta de GitHub

<https://help.github.com/en/articles/signing-up-for-a-new-github-account>

apéndice C

Creación de una cuenta Quay

Meta

Describir cómo crear una cuenta de Quay para los trabajos de laboratorio del curso.

Creación de una cuenta Quay

Objetivos

Tras finalizar esta sección, deberá ser capaz de crear una cuenta de Quay y crear repositorios de imágenes de contenedores públicos para los trabajos de laboratorio del curso.

Creación de una cuenta Quay

Necesita una cuenta de Quay para crear uno o más repositorios de imágenes de contenedores *públicos* para los ejercicios de laboratorio de este curso. Si ya tiene una cuenta Quay, puede omitir los pasos para crear una nueva cuenta que se detalla en este apéndice.



Importante

Si ya tiene una cuenta de Quay, asegúrese de crear solo repositorios de imágenes de contenedores *públicos* para los trabajos de laboratorio de este curso. Las instrucciones y los scripts de evaluación del trabajo de laboratorio requieren acceso no autenticado para extraer imágenes de contenedores del repositorio.

Para crear una nueva cuenta Quay, realice los siguientes pasos:

1. Diríjase a <https://quay.io> mediante un navegador web.
2. Haga clic en **Sign in** (Inicio de sesión) en la esquina superior derecha (junto a la barra de búsqueda).
3. En la página **Sign in** (Inicio de sesión), puede iniciar sesión con sus credenciales de Google o GitHub (creadas en el Apéndice A).

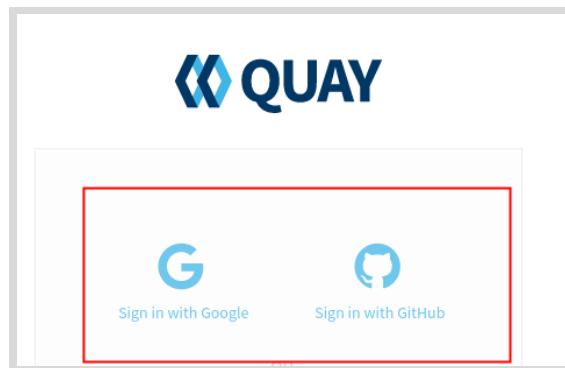


Figura C.1: Inicie sesión con las credenciales de Google o GitHub.

De manera alternativa, haga clic en **Create account** (Crear cuenta) para crear una nueva cuenta.

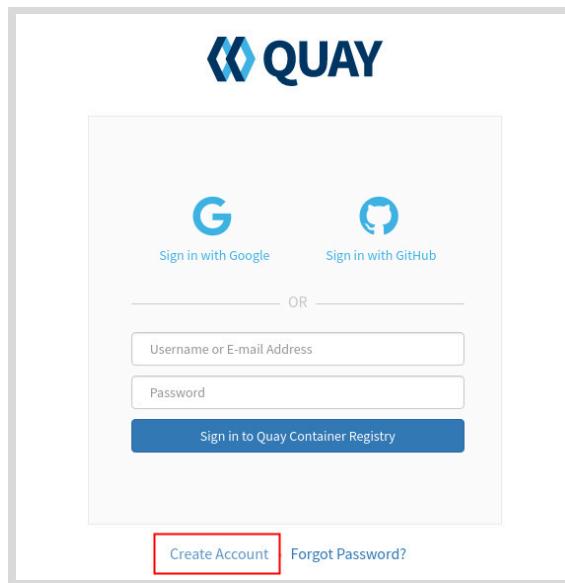


Figura C.2: Creación de una nueva cuenta

4. Si optó por omitir el método de inicio de sesión en Google o GitHub y, en cambio, optó por crear una nueva cuenta, recibirá un correo electrónico con instrucciones sobre cómo activar su cuenta de Quay. Verifique su dirección de correo electrónico y, luego, inicie sesión en el sitio web de Quay con el nombre de usuario y la contraseña que proporcionó durante la creación de la cuenta.
5. Después de haber iniciado sesión en Quay, puede crear nuevos repositorios de imágenes con un clic en **Create New Repository** (Crear nuevo repositorio) en la página **Repositories** (Repositorios).

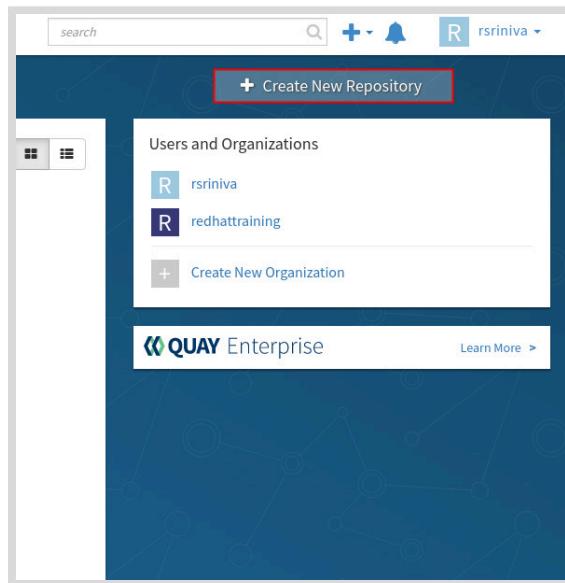


Figura C.3: Creación de un nuevo repositorio de imágenes

De manera alternativa, haga clic en el ícono con el signo más (+) en la esquina superior derecha (a la izquierda del ícono de campana) y, luego, haga clic en **New Repository** (Nuevo repositorio).

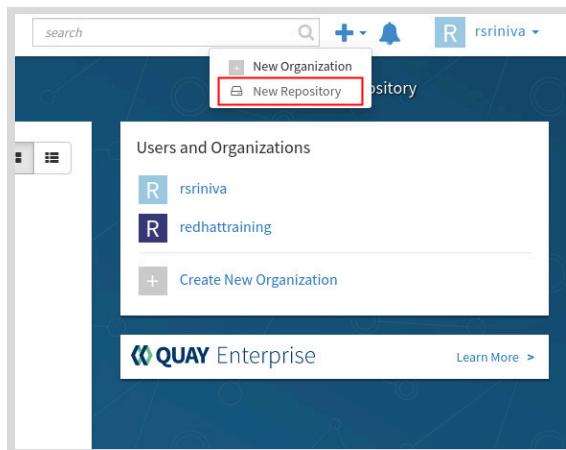


Figura C.4: Creación de un nuevo repositorio de imágenes



Referencias

Comencemos con Quay.io

<https://docs.quay.io/solution/getting-started.html>

Visibilidad de repositorios

Objetivos

Tras finalizar esta sección, deberá ser capaz de controlar la visibilidad de repositorios en Quay.io.

Visibilidad de repositorios en Quay.io

Quay.io ofrece la posibilidad de crear repositorios públicos y privados. Cualquier persona puede leer los repositorios públicos sin restricciones, a pesar de que los permisos de escritura se deben otorgar de manera explícita. Los repositorios privados tienen permisos de lectura y escritura restringidos. Sin embargo, la cantidad de repositorios privados en quay.io es limitada según el plan del espacio de nombres.

Visibilidad predeterminada del repositorio

De forma predeterminada, los repositorios creados mediante el envío de imágenes a quay.io son privados. Para que OpenShift (o cualquier otra herramienta) obtenga esas imágenes, puede configurar una clave privada en OpenShift y Quay o hacer que el repositorio sea público, de modo que no se requiera autenticación. La configuración de claves privadas está fuera del alcance de este documento.

The screenshot shows the Red Hat Quay.io interface. At the top, there is a navigation bar with icons for back, forward, and refresh, followed by a URL field containing <https://quay.io/repository/>. Below the URL is the Red Hat Quay.io logo. To the right of the logo are menu items: EXPLORE, APPLICATIONS, REPOSITORIES (which is highlighted in red), and TUTORIAL. The main content area has a dark background with a network-like graphic. The title "Repositories" is centered at the top of this area. On the left, under the heading "Starred", it says "You haven't starred any repositories yet." and "Stars allow you to easily access your favorite repositories." Below this, there is a section titled "RHT_OCP4_QUAY_USER" which lists four repositories: "do180-mysql-57-rhel7", "do180-todonodejs", "do180-quote-php", and "nexus". Each repository entry includes a small icon, the repository name, and a star icon.

Actualización de la visibilidad del repositorio

Para establecer la visibilidad del repositorio como público, seleccione el repositorio correspondiente en <https://quay.io/repository/> (inicie sesión en su cuenta si es necesario) y haga clic en el engranaje ubicado en el borde inferior izquierdo para abrir la página **Settings** (Configuración). Desplácese hacia abajo hasta la sección **Repository Visibility** (Visibilidad de repositorios) y haga clic en el botón **Make Public** (Hacer público).

The screenshot shows the Quay.io repository settings interface. At the top, it says "Trust Disabled" with a gear icon and a button to "Enable Trust". Below that is the "Events and Notifications" section, which states "No notifications have been setup for this repository." and has a "Create Notification" button. The "Repository Visibility" section shows a lock icon and notes that the repository is currently private, with a "Make Public" button. The "Delete Repository" section contains a warning message: "Deleting a Repository cannot be undone. Here be dragons!" and a red "Delete Repository" button.

Vuelva a la lista de repositorios. El ícono del candado junto al nombre de repositorio ha desaparecido, lo que indica que el repositorio se ha hecho público.

apéndice D

Comandos Git útiles

Meta

Describir comandos Git útiles que se usan para los trabajos de laboratorio de este curso.

Comandos Git

Objetivos

Tras finalizar esta sección, deberá ser capaz de reiniciar y rehacer ejercicios en este curso. También debe ser capaz de cambiar de un ejercicio incompleto a otro y, luego, continuar con el ejercicio anterior en el lugar en que lo dejó.

Trabajo con bifurcaciones Git

En este curso se usa un repositorio Git alojado en GitHub para almacenar el código fuente del código de curso de la aplicación. Al principio del curso, crea su propia bifurcación de este repositorio, que también se aloja en GitHub.

Durante este curso, trabaja con una copia local de la bifurcación, que clona en la máquina virtual **workstation**. El término *origen* hace referencia al repositorio remoto desde el que se clona un repositorio local.

Al trabajar con los ejercicios del curso, usa diferentes bifurcaciones Git para cada ejercicio. Todos los cambios que realice en el código fuente ocurren en una nueva bifurcación que solo crea para ese ejercicio. Nunca realice cambios en la bifurcación **master**.

A continuación, se incluye una lista de escenarios y los comandos Git correspondientes que puede usar para trabajar con bifurcaciones, y para regresar a un estado bueno conocido.

Rehacer un ejercicio desde cero

Para rehacer un ejercicio desde cero después de completarlo, realice los siguientes pasos:

- Usted confirma y envía todos los cambios en su bifurcación local como parte de la realización del ejercicio. Para finalizar el ejercicio, ejecute su subcomando **finish** para limpiar todos los recursos:

```
[student@workstation ~]$ lab your-exercise finish
```

- Cambie a su clon local del repositorio **D0180-apps** y cambie a la bifurcación **master**:

```
[student@workstation ~]$ cd ~/D0180-apps  
[student@workstation D0180-apps]$ git checkout master
```

- Elimine su bifurcación local:

```
[student@workstation D0180-apps]$ git branch -d your-branch
```

- Elimine la bifurcación remota en su cuenta de GitHub personal:

```
[student@workstation D0180-apps]$ git push origin --delete your-branch
```

- Use el subcomando **start** para reiniciar el ejercicio:

```
[student@workstation D0180-apps]$ cd ~  
[student@workstation ~]$ lab your-exercise start
```

Abandono de un ejercicio parcialmente completado y reinicio de este desde cero

Puede encontrarse en un escenario en el que haya completado parcialmente algunos pasos en el ejercicio y desee abandonar el intento actual y reiniciarlo desde cero. Realice los siguientes pasos:

1. Ejecute el subcomando **finish** del ejercicio para limpiar todos los recursos.

```
[student@workstation ~]$ lab your-exercise finish
```

2. Ingrese su clon local del repositorio **D0180-apps** y descarte los cambios pendientes en la bifurcación actual mediante **git stash**:

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

3. Cambie a la bifurcación **maestra** de su repositorio local:

```
[student@workstation D0180-apps]$ git checkout master
```

4. Elimine su bifurcación local:

```
[student@workstation D0180-apps]$ git branch -d your-branch
```

5. Elimine la bifurcación remota en su cuenta de GitHub personal:

```
[student@workstation D0180-apps]$ git push origin --delete your-branch
```

6. Ahora puede reiniciar el ejercicio ejecutando su subcomando **start**:

```
[student@workstation D0180-apps]$ cd ~  
[student@workstation ~]$ lab your-exercise start
```

Cambio a un ejercicio diferente desde un ejercicio incompleto

Puede encontrarse en un escenario en el que haya completado parcialmente algunos pasos en un ejercicio, pero desee cambiar a un ejercicio diferente y volver a revisar el ejercicio actual en un momento posterior.

Evite dejar incompletos muchos ejercicios para volver allí más tarde. Estos ejercicios componen los recursos de la nube, y puede usar la cuota asignada en el proveedor de la nube y en el clúster OpenShift que comparte con otros estudiantes. Si cree que puede transcurrir un tiempo hasta que pueda volver al ejercicio actual, considere la posibilidad de abandonarlo y, luego, reiniciarlo desde cero.

Si prefiere hacer una pausa en el ejercicio actual y trabajar en el siguiente, realice los siguientes pasos:

apéndice D | Comandos Git útiles

1. Confirme los cambios pendientes en su repositorio local y envíelos a su cuenta de GitHub personal. Es posible que desee registrar el paso en el que detuvo el ejercicio:

```
[student@workstation ~]$ cd ~/D0180-apps  
[student@workstation D0180-apps]$ git commit -a -m 'Paused at step X.Y'  
[student@workstation D0180-apps]$ git push
```

2. No ejecute el comando **finish** del ejercicio original. Esto es importante para dejar los proyectos de OpenShift existentes sin modificaciones, de modo que puede reanudarlos más adelante.
3. Comience el siguiente ejercicio ejecutando su subcomando **start**:

```
[student@workstation ~]$ lab your-exercise start
```

4. El siguiente ejercicio cambia a la bifurcación **master** y, opcionalmente, crea una nueva bifurcación para sus cambios. Esto significa que los cambios realizados en el ejercicio original en la bifurcación original se dejan sin tocar.
5. Más adelante, una vez que haya completado el siguiente ejercicio, y desee volver al ejercicio original, vuelva a su bifurcación:

```
[student@workstation ~]$ git checkout original-branch
```

Más adelante, puede continuar con el ejercicio original en el paso donde lo dejó.



Referencias

Página del manual de la bifurcación Git

<https://git-scm.com/docs/git-branch>

¿Qué es una bifurcación Git?

<https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell>

Herramientas Git - Almacenamiento

<https://git-scm.com/book/en/v2/Git-Tools-Stashing-and-Cleaning>