



Red Hat OpenShift Development II: Containerizing Applications



OCP 4.5 DO288
Red Hat OpenShift Development II: Containerizing
Applications
Edición 3 20200928
fecha de publicación 20200928

Autores: Zach Guterman, Richard Allred, Ricardo Jun,
Ravishankar Srinivasan, Fernando Lozano, Ivan Chavero,
Dan Kolepp, Jordi Sola Alaball, Manuel Aude Morales,
Eduardo Ramírez Martínez
Editor: David O'Brien, Seth Kenlon

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: Grega Bremec, Sajith Sugathan, Dave Sacco, Rob Locke, Bowe Strickland, Rudolf Kastl, Chris Tusa

| | |
|--|------------|
| Convenciones del documento | ix |
| Introducción | xi |
| Red Hat OpenShift Development II: Containerizing Applications | xii |
| Orientación sobre el entorno del aula | xiii |
| Internacionalización | xvi |
| 1. Implementación y administración de aplicaciones en un clúster OpenShift | 1 |
| Presentación de OpenShift Container Platform 4 | 2 |
| Cuestionario: Presentación de OpenShift 4 | 8 |
| Ejercicio Guiado: Configuración del entorno del aula | 12 |
| Implementación de una aplicación en un clúster OpenShift | 18 |
| Ejercicio Guiado: Implementación de una aplicación en un clúster OpenShift | 27 |
| Administración de aplicaciones con la consola web | 35 |
| Ejercicio Guiado: Administración de una aplicación con la consola web | 40 |
| Administración de aplicaciones con la CLI | 49 |
| Ejercicio Guiado: Administración de una aplicación con la CLI | 54 |
| Trabajo de laboratorio: Implementación y administración de aplicaciones en un clúster OpenShift | 64 |
| Resumen | 73 |
| 2. Diseño de aplicaciones contenerizadas para OpenShift | 75 |
| Selección de un enfoque de contenerización | 76 |
| Cuestionario: Selección de un enfoque de contenerización | 80 |
| Compilación de imágenes de contenedor con instrucciones avanzadas de Dockerfile | 86 |
| Ejercicio Guiado: Compilación de imágenes de contenedor con instrucciones avanzadas de Dockerfile | 94 |
| Inserción de datos de configuración en una aplicación | 102 |
| Ejercicio Guiado: Inserción de datos de configuración en una aplicación | 110 |
| Trabajo de laboratorio: Diseño de aplicaciones contenerizadas para OpenShift | 118 |
| Resumen | 128 |
| 3. Publicación de imágenes de contenedor empresariales | 129 |
| Administración de imágenes en un registro empresarial | 130 |
| Ejercicio Guiado: Uso de un registro empresarial | 139 |
| Permiso de acceso al registro de OpenShift | 145 |
| Ejercicio Guiado: Uso del registro de OpenShift | 149 |
| Creación de flujos de imágenes | 153 |
| Ejercicio Guiado: Creación de un flujo de imagen | 161 |
| Trabajo de laboratorio: Publicación de imágenes de contenedor empresariales | 165 |
| Resumen | 174 |
| 4. Compilación de aplicaciones | 175 |
| Descripción del proceso de compilación de OpenShift | 176 |
| Cuestionario: El proceso de compilación de OpenShift | 180 |
| Administración de compilaciones de una aplicación | 186 |
| Ejercicio Guiado: Administración de compilaciones de una aplicación | 190 |
| Desencadenamiento de compilaciones | 196 |
| Ejercicio Guiado: Desencadenamiento de compilaciones | 199 |
| Implementación de hooks de compilación post-commit (posteriores a la confirmación) .. | 206 |
| Ejercicio Guiado: Implementación de hooks de compilación post-commit (posteriores a la confirmación) | 208 |
| Trabajo de laboratorio: Compilación de aplicaciones | 213 |
| Resumen | 222 |
| 5. Personalización de compilaciones de fuente a imagen | 223 |
| Descripción de la arquitectura de fuente a imagen | 224 |
| Cuestionario: Descripción de la arquitectura de fuente a imagen | 230 |

| | |
|--|------------|
| Personalización de una imagen base S2I existente | 234 |
| Ejercicio Guiado: Personalización de compilaciones S2I | 237 |
| Creación de una imagen de compilador S2I | 242 |
| Ejercicio Guiado: Creación de una imagen de compilador S2I | 248 |
| Trabajo de laboratorio: Personalización de compilaciones de fuente a imagen | 259 |
| Resumen | 275 |
| 6. Creación de aplicaciones desde plantillas de OpenShift | 277 |
| Descripción de los elementos de una plantilla de OpenShift | 278 |
| Cuestionario: Descripción de los elementos de una plantilla de OpenShift | 283 |
| Creación de una plantilla con varios contenedores | 285 |
| Ejercicio Guiado: Creación de una plantilla con varios contenedores | 290 |
| Trabajo de laboratorio: Creación de aplicaciones desde plantillas de OpenShift | 301 |
| Resumen | 312 |
| 7. Administración de las implementaciones de aplicaciones | 313 |
| Monitoreo del estado de la aplicación | 314 |
| Ejercicio Guiado: Activación de sondeos | 319 |
| Selección de la estrategia de implementación adecuada | 325 |
| Ejercicio Guiado: Aplicación de una estrategia de implementación | 329 |
| Administración de implementaciones de aplicaciones con Comandos CLI | 337 |
| Ejercicio Guiado: Administración de las implementaciones de aplicaciones | 342 |
| Trabajo de laboratorio: Administración de las implementaciones de aplicaciones | 349 |
| Resumen | 361 |
| 8. Implementación de tuberías de integración continua y de implementación continua en OpenShift | 363 |
| Descripción de conceptos de CI/CD | 364 |
| Cuestionario: Conceptos de CI/CD | 368 |
| Implementación de tuberías de Jenkins en OpenShift | 372 |
| Ejercicio Guiado: Ejecutar una tubería simple de Jenkins | 378 |
| Escritura de tuberías de Jenkins personalizadas | 390 |
| Ejercicio Guiado: Crear y ejecutar una tubería de Jenkins | 395 |
| Trabajo de laboratorio: Implementación de canalizaciones de integración continua y de implementación continua en OpenShift | 408 |
| Resumen | 426 |
| 9. Compilación de aplicaciones para OpenShift | 427 |
| Integración de servicios externos | 428 |
| Ejercicio Guiado: Integración de un servicio externo | 431 |
| Servicios de contenedorización | 435 |
| Ejercicio Guiado: Implementación de un servidor Nexus contenedorizado | 443 |
| Implementación de aplicaciones con Red Hat OpenShift Application Runtimes | 453 |
| Ejercicio Guiado: Implementación de una aplicación con Red Hat OpenShift Application Runtimes | 460 |
| Trabajo de laboratorio: Compilación de aplicaciones nativas de la nube para OpenShift ... | 470 |
| Resumen | 480 |
| 10. Revisión completa: Red Hat OpenShift Development II: Containerizing Applications | 481 |
| Revisión exhaustiva | 482 |
| Trabajo de laboratorio: Diseño de una imagen de contenedor para OpenShift | 485 |
| Trabajo de laboratorio: Contenerización e implementación de un servicio | 496 |
| Trabajo de laboratorio: Compilación e implementación de una aplicación con varios contenedores | 509 |
| A. Creación de una cuenta GitHub | 527 |
| Creación de una cuenta GitHub | 528 |

| | |
|---------------------------------------|------------|
| B. Creación de una cuenta Quay | 531 |
| Creación de una cuenta Quay | 532 |
| C. Comandos Git útiles | 535 |
| Comandos Git | 536 |

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. Omitir una nota no debería tener consecuencias negativas, pero quizás se pase 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

Red Hat OpenShift Development II: Containerizing Applications

Red Hat OpenShift Container Platform, basada en tecnología de contenedores y Kubernetes, brinda a los desarrolladores una solución de entorno empresarial para desarrollar e implementar aplicaciones de software contenedorizadas.

Red Hat OpenShift Development I: Containerizing Applications (DO288), el segundo curso del recorrido de desarrollo de OpenShift, enseña a los estudiantes a diseñar, compilar e implementar aplicaciones de software contenedorizado en un clúster OpenShift. Sin importar si crea aplicaciones nativas en contenedores o migra desde aplicaciones existentes, este curso le brinda capacitación práctica para aumentar la productividad del desarrollador mediante Red Hat OpenShift Container Platform.

Objetivos del curso

- Diseñar, compilar e implementar aplicaciones contenedorizadas en un clúster OpenShift.

Destinatarios

- Desarrolladores de software
- Arquitectos de software

Requisitos previos

- Haber completado el curso *Introduction to Containers, Kubernetes, and Red Hat OpenShift* (DO180) o tener conocimientos equivalentes.
- La certificación RHCSA, o superior, es útil para la navegación y el uso de la línea de comandos, pero no es un requisito.

Orientación sobre el entorno del aula

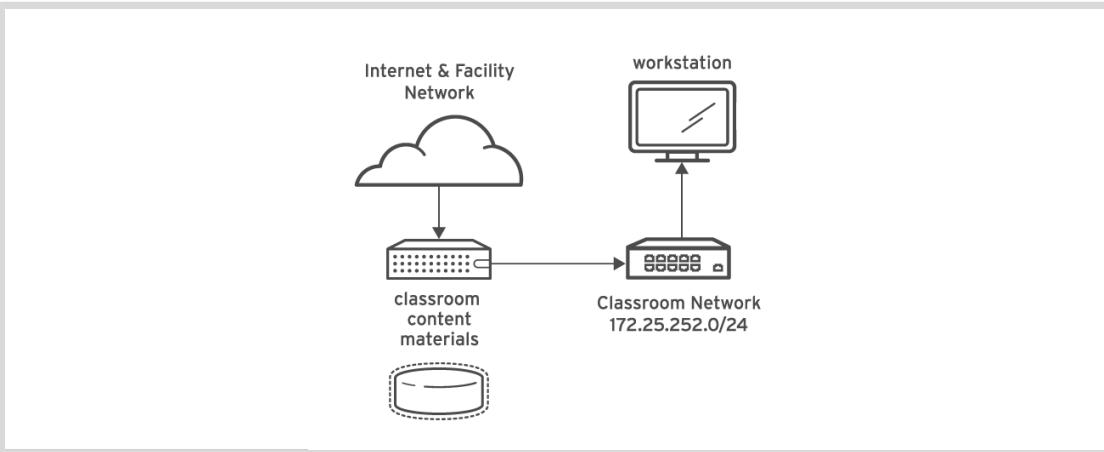


Figura 0.1: Entorno del aula

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 |
|--|--|--|
| <code>content.example.com</code> , <code>materials.example.com</code> , <code>classroom.example.com</code> | 172.25.252.254, 172.25.253.254, 172.25.254.254 | Servidor de utilidades del aula |
| <code>workstation.lab.example.com</code> | 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 DO288.

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 DO288 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 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 <code>workstation</code> y usar <code>ssh</code> para conectarse a las otras máquinas virtuales. |
| ACTION (ACCIÓN) → Start(Iniciar) | Inicia (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 otorga 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 (Región e idioma).

Puede iniciar esta aplicación de dos maneras. Puede ejecutar el comando `gnome-control-center region` desde una ventana de terminal, o en la barra superior, desde el menú del sistema en la esquina derecha, seleccionar el botón de configuración (que tiene un ícono de un destornillador y una llave cruzados) desde 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 usarán 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** (Fuentes de entrada). Se abrirá la ventana **Add an Input Source** (Aregar una fuente de entrada). 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 iconos de 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, en **English (United States)** (Inglés [Estados Unidos]) está la distribución de 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**

Puede ingresar cualquier carácter Unicode en el entorno de escritorio GNOME si conoce el código Unicode del carácter. Ingrese **Ctrl+Shift+U**, seguido por el código. Después de ingresar **Ctrl+Shift+U**, aparece una u subrayada que indica 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, **Intro**.

Valores de idioma predeterminados 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 tendrá efecto para usuarios en su próximo 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 disponibles se pueden configurar a través de `localectl` tanto para las consolas virtuales de texto como para el 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 --what supplements` para determinar qué paquetes de RPM pueden ser instalados por un paquete de idiomas:

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

**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 tesauro *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 |
| Punjabí | 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

Implementación y administración de aplicaciones en un clúster OpenShift

Meta

Implementar aplicaciones mediante diferentes métodos de empaquetado de aplicaciones en un clúster OpenShift y administrar sus recursos.

Objetivos

- Describir la arquitectura y las nuevas características de OpenShift 4.
- Implementar una aplicación en el clúster desde un Dockerfile con la CLI.
- Implementar una aplicación desde una imagen de contenedor y administrar sus recursos mediante la consola web.
- Implementar una aplicación desde el código fuente y administrar sus recursos mediante la interfaz de línea de comandos.

Secciones

- Presentación de OpenShift 4 (y cuestionario)
- Implementación de una aplicación en un clúster OpenShift (y ejercicio guiado)
- Administración de aplicaciones con la consola web (y ejercicio guiado)
- Administración de aplicaciones con la CLI (y ejercicio guiado)

Trabajo de laboratorio

Implementación y administración de aplicaciones en un clúster OpenShift

Presentación de OpenShift Container Platform 4

Objetivos

Después de completar esta sección, deberá ser capaz de describir la arquitectura y las nuevas características en OpenShift Container Platform 4.

Arquitectura de OpenShift Container Platform 4

Red Hat OpenShift Container Platform 4 (RHOC 4) es un conjunto de componentes y servicios modulares desarrollados sobre la base de Red Hat CoreOS y Kubernetes. OpenShift agrega capacidades de *plataforma como servicio (PaaS)*, como administración remota, mayor seguridad, monitoreo y auditoría, administración del ciclo de vida de la aplicación e interfaces de autoservicio para desarrolladores. Proporciona servicios de organización y simplifica la implementación, la administración y el escalamiento de las aplicaciones contenedorizadas.

Un clúster OpenShift se puede administrar de la misma forma que cualquier otro clúster Kubernetes y también usando las herramientas de administración provistas por OpenShift, como la interfaz de línea de comandos o la consola web. Estas herramientas adicionales permiten flujos de trabajo más productivos y hacen que las tareas cotidianas sean mucho más fáciles de administrar.

Una de las principales ventajas de usar OpenShift es que usa varios nodos para garantizar la flexibilidad y la escalabilidad de sus aplicaciones administradas. OpenShift forma un clúster de servidores de nodos que ejecutan contenedores y se administran de forma central mediante un conjunto de servidores maestros. Un solo servidor puede actuar como maestro y como nodo, pero generalmente deberían ir separados para brindar mayor estabilidad y alta disponibilidad.

En el siguiente diagrama, se ilustra la visión general lógica de alto nivel de la arquitectura de OpenShift Container Platform 4.

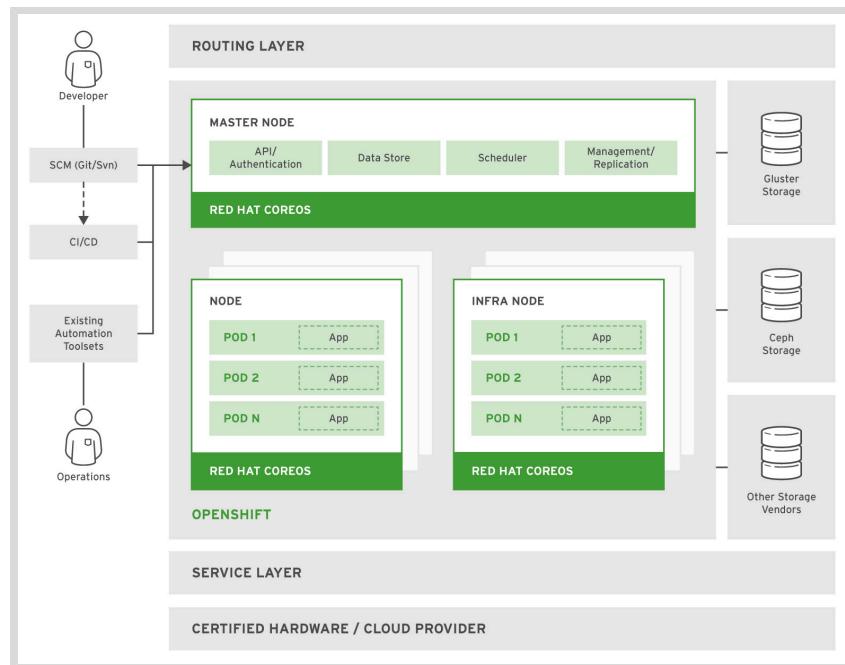


Figura 1.1: Arquitectura de OpenShift 4

En el siguiente diagrama, se ilustra la pila (stack) de OpenShift Container Platform.

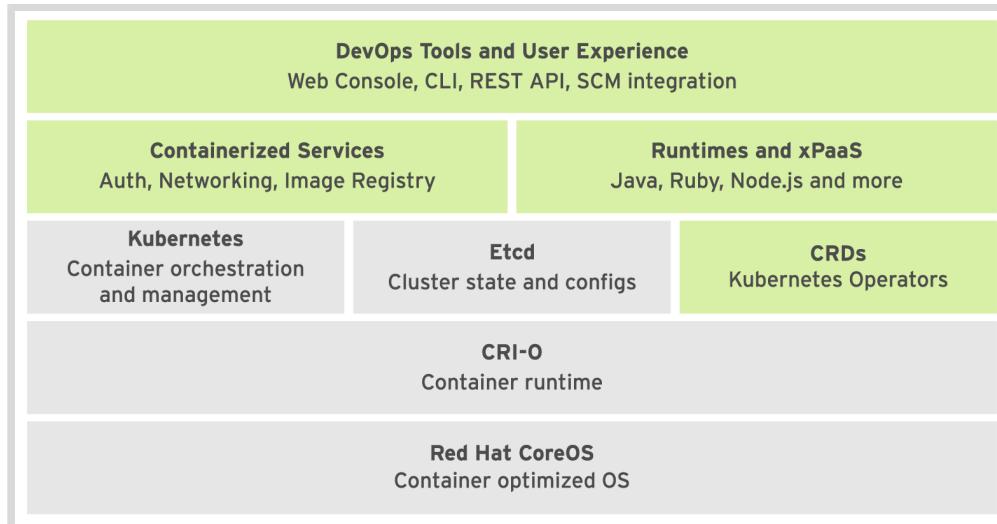


Figura 1.2: Pila (stack) de componentes OpenShift

Desde abajo hacia arriba y de izquierda a derecha, se muestra la infraestructura de contenedores básica, integrada y mejorada por Red Hat:

Red Hat CoreOS

Red Hat CoreOS es el sistema operativo base en la parte superior que ejecuta OpenShift.

Red Hat CoreOS es una distribución de Linux enfocada en proporcionar un sistema operativo inmutable para la ejecución de contenedores.

CRI-O

CRI-O es una implementación del Kubernetes Container Runtime Interface, interfaz de tiempo de ejecución del contenedor (CRI) para habilitar el uso de tiempos de ejecución compatibles con Open Container Initiative, iniciativa de contenedor abierto (OCI). CRI-O puede usar el

tiempo de ejecución de cualquier contenedor que satisfaga la CRI, por ejemplo: runc (usado por el servicio Docker) o rkt (desde CoreOS).

Kubernetes

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.

Etcdb

Etcdb es un almacén de clave–valor distribuido, usado por Kubernetes para almacenar información de estado y configuración acerca de los contenedores y otros recursos dentro del clúster de Kubernetes.

Definiciones de recursos personalizados (CRD)

Las *definiciones de recursos personalizados (CRD)* son tipos de recursos almacenados en Etcdb y administrados por Kubernetes. Estos tipos de recursos forman el estado y la configuración de todos los recursos administrados por OpenShift.

Servicios contenedrizados

Los servicios contenedrizados cumplen muchas funciones de infraestructura de PaaS, como redes y autorización. RHOCP 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 RHOCP se ejecutan como contenedores orquestados por Kubernetes.

Tiempos de ejecución y xPaaS

Tiempos de ejecución y xPaaS: son imágenes de contenedor 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

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.

En la siguiente tabla, se enumeran algunas de las terminologías usadas con mayor frecuencia cuando trabaja con OpenShift.

Terminología de OpenShift

| Término | Definición |
|-------------|--|
| Nodo | Un servidor que aloja aplicaciones en un clúster OpenShift. |
| Nodo master | Un servidor de nodo que administra el plano de control en un clúster OpenShift. Los nodos master proporcionan servicios de clúster básicos, como API o controladores. |

| Término | Definición |
|-------------------------------|--|
| Nodo trabajador | También denominado nodo de cómputo, los nodos trabajador ejecutan cargas de trabajo para el clúster. Los pods de la aplicación se programan en los nodos trabajadores. |
| Recurso | Los recursos son cualquier tipo de definición de componentes administrada por OpenShift. 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 componente de OpenShift 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 OpenShift. Los selectores usan etiquetas con el fin de filtrar recursos elegibles para programaciones y otras operaciones. |
| Espacio de nombres o proyecto | Un alcance para los recursos y procesos de OpenShift, de modo que los recursos con el mismo nombre se puedan usar en diferentes contextos. |
| Consola | Interfaz de usuario web proporcionada por OpenShift que permite a los desarrolladores y administradores administrar los recursos del clúster. |

**nota**

Las últimas versiones de OpenShift 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 Operator Framework están fuera del alcance de este curso.

Nuevas características en RHOPC 4

RHOPC 4 es un cambio significativo de versiones anteriores. Además de mantener la compatibilidad con versiones anteriores, incluye nuevas características, como:

- CoreOS como sistema operativo predeterminado para todos los nodos, que ofrece una infraestructura inmutable optimizada para contenedores.
- Un nuevo instalador de clúster, que simplifica el proceso de instalación y actualización de los nodos master y trabajador en el clúster.
- Una plataforma que se administra de forma automática, capaz de aplicar automáticamente las actualizaciones y recuperaciones de clústeres sin interrupciones.
- Una consola web rediseñada basada en el concepto de "personas", dirigida tanto a administradores de plataformas como a desarrolladores.
- Un SDK del operador para compilar, probar y empaquetar operadores.

Descripción de tipos de recursos de OpenShift

Como desarrollador, trabajará con muchas clases diferentes de tipos de recursos en OpenShift. Estos recursos que pueden crearse y configurarse con un archivo YAML o JSON, o con herramientas de administración de OpenShift:

Pods (pod)

Colecciones de contenedores que comparten recursos, como direcciones IP y volúmenes de almacenamiento persistente. Los pods son la unidad básica de trabajo para OpenShift.

Servicios (svc)

Combinaciones específicas de IP/puerto que proporcionan acceso a un conjunto (pool) de pods. De forma predeterminada, los servicios conectan clientes con pods con el método Round Robin.

Controladores de replicación (rc)

Recursos de OpenShift que definen cómo se replican los pods (escalado horizontalmente) en diferentes nodos. Los controladores de replicación son un servicio básico de OpenShift para proporcionar una alta disponibilidad para pods y contenedores.

Volúmenes persistentes (pv)

Áreas de almacenamiento que usarán los pods.

Reclamaciones de volumen persistente (pvc)

Solicitudes 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.

Mapas de config. (cm)

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 Secrets se usa para almacenar datos sensibles (normalmente cifrados) y su acceso está restringido a menos usuarios autorizados.

Configuraciones de implementación (dc)

Un 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.

Configuraciones de compilación (bc)

Un proceso que se ejecutará en el proyecto OpenShift. La característica de fuente a imagen (*source-to-image [S2I]*) de OpenShift usa BuildConfigs para compilar una imagen de contenedor a partir del código fuente de la aplicación almacenado en un repositorio Git. Una bc funciona junto con una dc para proporcionar flujos de trabajo de entrega continua e integración continua básicos pero ampliables.

Rutas

Nombres de host de DNS reconocidos por el enrutador de OpenShift como un punto de entrada para diversas aplicaciones y microservicios implementados en el clúster.

Flujos de imágenes (es)

Un flujo de imágenes y sus etiquetas proporcionan una abstracción para hacer referencia a imágenes de contenedor desde OpenShift Container Platform. El flujo de imágenes y sus etiquetas le permiten realizar un seguimiento de las imágenes disponibles y garantizar que esté usando la imagen específica que necesita, aun si cambia la imagen en el repositorio. Los flujos de imágenes no contienen datos de imágenes reales, pero presentan una vista virtual individual de imágenes relacionadas, similar a un repositorio de imágenes.



Referencias

Sitio web de documentación de Kubernetes

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

Sitio web de documentación de OpenShift

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

Operadores CoreOS y Operator Framework

<https://coreos.com/operators/>

► Cuestionario

Presentación de OpenShift 4

Elija las respuestas correctas para las siguientes preguntas:

Una vez que haya finalizado el cuestionario, haga clic en **CHECK** (VERIFICAR). Si quiere volver a intentarlo, haga clic en **RESET** (RESTABLECER). Haga clic en **SHOW SOLUTION** (MOSTRAR SOLUCIÓN) para ver todas las respuestas correctas.

► 1. **¿Cuál es el enunciado correcto acerca de las adiciones de OpenShift en relación con Kubernetes?**

- a. OpenShift agrega características para que el desarrollo y la implementación de aplicaciones en Kubernetes sean fáciles y eficientes.
- b. Las imágenes de contenedores creadas para OpenShift no pueden usarse con Kubernetes simple.
- c. Para habilitar las nuevas características, Red Hat mantiene versiones bifurcadas de Kubernetes internas al producto de RHOP.
- d. No hay nuevas características para la integración continua y la implementación continua con RHOP, pero puede usar herramientas externas en su lugar.

► 2. **¿Cuál es el enunciado correcto acerca del almacenamiento persistente en OpenShift?**

- a. Los desarrolladores crean una reclamación de volumen persistente para solicitar un área de almacenamiento del clúster que un pod de proyecto puede usar para almacenar datos.
- b. Una reclamación de volumen persistente representa un área de almacenamiento que un pod puede solicitar para almacenar datos, pero es aprovisionada por el administrador del clúster.
- c. Una reclamación de volumen persistente 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. Una reclamación de volumen persistente 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.
- e. OpenShift admite almacenamiento persistente, ya que permite a los administradores asignar directamente el almacenamiento disponible en los nodos a las aplicaciones que se ejecutan en el clúster.

► **3. ¿Cuáles son las dos afirmaciones correctas acerca de los tipos de recursos de 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. Un servicio es responsable de proporcionar direcciones IP para el acceso externo a los pods.
- d. Una ruta es responsable de proporcionar un nombre de host para el acceso externo a los pods.
- e. Un controlador de replicación es responsable de monitorear y mantener la cantidad de pods desde una aplicación específica.

► **4. ¿Cuáles son las dos afirmaciones correctas acerca de la arquitectura de OpenShift 4? (Elija dos opciones).**

- a. Los nodos de OpenShift pueden administrarse sin un maestro. Los nodos forman una red de par a par.
- b. Los nodos maestros de OpenShift gestionan el escalamiento y la programación de pods que se ejecutan en los nodos.
- c. Los nodos master de un clúster deben ejecutar Red Hat CoreOS.
- d. Los nodos master de un clúster deben ejecutar Red Hat Enterprise Linux 8.
- e. Los nodos maestros de un clúster deben ejecutar Red Hat Enterprise Linux 7.

► Solución

Presentación de OpenShift 4

Elija las respuestas correctas para las siguientes preguntas:

Una vez que haya finalizado el cuestionario, haga clic en **CHECK** (VERIFICAR). Si quiere volver a intentarlo, haga clic en **RESET** (RESTABLECER). Haga clic en **SHOW SOLUTION** (MOSTRAR SOLUCIÓN) para ver todas las respuestas correctas.

► 1. ¿Cuál es el enunciado correcto acerca de las adiciones de OpenShift en relación con Kubernetes?

- a. OpenShift agrega características para que el desarrollo y la implementación de aplicaciones en Kubernetes sean fáciles y eficientes.
- b. Las imágenes de contenedores creadas para OpenShift no pueden usarse con Kubernetes simple.
- c. Para habilitar las nuevas características, Red Hat mantiene versiones bifurcadas de Kubernetes internas al producto de RHOP.
- d. No hay nuevas características para la integración continua y la implementación continua con RHOP, pero puede usar herramientas externas en su lugar.

► 2. ¿Cuál es el enunciado correcto acerca del almacenamiento persistente en OpenShift?

- a. Los desarrolladores crean una reclamación de volumen persistente para solicitar un área de almacenamiento del clúster que un pod de proyecto puede usar para almacenar datos.
- b. Una reclamación de volumen persistente representa un área de almacenamiento que un pod puede solicitar para almacenar datos, pero es aprovisionada por el administrador del clúster.
- c. Una reclamación de volumen persistente 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. Una reclamación de volumen persistente 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.
- e. OpenShift admite almacenamiento persistente, ya que permite a los administradores asignar directamente el almacenamiento disponible en los nodos a las aplicaciones que se ejecutan en el clúster.

► **3. ¿Cuáles son las dos afirmaciones correctas acerca de los tipos de recursos de 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. Un servicio es responsable de proporcionar direcciones IP para el acceso externo a los pods.
- d. Una ruta es responsable de proporcionar un nombre de host para el acceso externo a los pods.
- e. Un controlador de replicación es responsable de monitorear y mantener la cantidad de pods desde una aplicación específica.

► **4. ¿Cuáles son las dos afirmaciones correctas acerca de la arquitectura de OpenShift 4? (Elija dos opciones).**

- a. Los nodos de OpenShift pueden administrarse sin un maestro. Los nodos forman una red de par a par.
- b. Los nodos maestros de OpenShift gestionan el escalamiento y la programación de pods que se ejecutan en los nodos.
- c. Los nodos master de un clúster deben ejecutar Red Hat CoreOS.
- d. Los nodos master de un clúster deben ejecutar Red Hat Enterprise Linux 8.
- e. Los nodos maestros de un clúster deben ejecutar Red Hat Enterprise Linux 7.

► 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 contenedor 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 DO288 en el entorno de aprendizaje en línea de la Capacitación Red Hat.
- 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 A, Creación de una cuenta GitHub*.
- Una cuenta personal y gratuita de Quay.io. Si necesita registrarse en Quay.io, consulte las instrucciones en *Apéndice B, 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, there's a section titled 'Lab Controls' with instructions to click 'CREATE' to build virtual machines and 'DELETE' to remove them. Below this are two buttons: 'DELETE' (red) and 'STOP' (blue). A blue information icon is also present. The main area displays 'OpenShift Details' with the following configuration:

| | | |
|-------------------------|---|---|
| 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, there are two rows for 'workstation' and 'classroom'. Each row has a status indicator ('active'), an 'ACTION' dropdown, and a green '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 contenerizadas.
- 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 las identificaciones, como nombres de host de ruta y nombres de proyecto, para evitar colisiones con identificadores de otros estudiantes que comparten el mismo clúster 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 A, *Creación de una cuenta GitHub* y Apéndice B, *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 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...
```

- 1.5. 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.
```

- 1.6. 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...
```

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

```
...output omitted...

Saving your lab configuration file...

Saving your Maven settings file...

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

- 1.8. 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/DO288-apps>. Si no inició sesión en GitHub, haga clic en **Sign in** (Inicio de sesión) en la esquina superior derecha.

The screenshot shows the GitHub interface for the repository 'RedHatTraining / DO288-apps'. At the top, there's a navigation bar with links for 'Why GitHub?', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. On the right side of the bar are 'Search', 'Sign in', and 'Sign up' buttons. Below the bar, the repository name 'RedHatTraining / DO288-apps' is displayed, along with a 'Code' button, 'Issues 0', 'Pull requests 0', 'Projects 0', 'Security', and 'Insights' buttons. To the right of the repository name are 'Watch 22', 'Star 0', 'Fork 3', and a 'Watch' dropdown menu.

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

The screenshot shows the GitHub sign-in page titled 'Sign in to GitHub'. It features a 'Username or email address' input field containing 'yourgituser', a 'Password' input field with masked text, and a 'Forgot password?' link. Below these fields is a green 'Sign in' button.

- 2.3. Regrese al repositorio RedHatTraining/D0288-apps y haga clic en Fork (Bifurcar) en la esquina superior derecha.

The screenshot shows the GitHub repository page for 'RedHatTraining / DO288-apps'. The same navigation bar and repository details are visible as in the previous screenshot. On the right side, there are 'Watch 22', 'Star 0', 'Fork 3', and a 'Watch' dropdown menu. The 'Fork' button is clearly visible.

- 2.4. En la ventana Fork DO288-apps, haga clic en yourgituser para seleccionar su proyecto de GitHub personal.

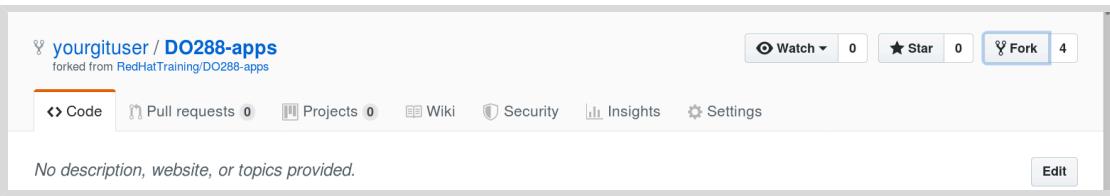
The screenshot shows a modal dialog box titled 'Fork DO288-apps'. The question 'Where should we fork DO288-apps?' is displayed. Below it, a list shows 'yourgituser' with a small purple user icon next to it. The background of the dialog is semi-transparent, showing parts of the GitHub interface.



Importante

Aunque es posible cambiar el nombre de la bifurcación personal del repositorio de <https://github.com/RedHatTraining/D0288-apps>, la calificación de scripts, scripts auxiliares y la salida de ejemplo de este curso suponen que conserva el nombre D0288-apps cuando bifurca el repositorio.

- 2.5. Despues de unos minutos, en la interfaz web de GitHub, se muestra su nuevo repositorio yourgituser/DO288-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/DO288-apps
Cloning into 'DO288-apps'...
...output omitted...
```

- 3.2. Verifique que /home/student/DO288-apps sea un repositorio Git:

```
[student@workstation ~]$ cd DO288-apps
[student@workstation DO288-apps]$ git status
# On branch master
nothing to commit, working directory clean
```

- 3.3. Verifique que /home/student/DO288-apps contenga las aplicaciones de muestra de este curso y vuelva a cambiar a la carpeta de inicio del usuario student.

```
[student@workstation DO288-apps]$ head README.md
# DO288 Containerized Example Applications
...output omitted...
[student@workstation DO288-apps]$ cd ~
[student@workstation ~]$
```

- 4. Ahora que tiene un clon local del repositorio DO288-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 código fuente se inician desde la bifurcación maestra del repositorio Git DO288-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 C, Comandos Git útiles*.

Esto concluye el ejercicio guiado.

Implementación de una aplicación en un clúster OpenShift

Objetivos

Tras finalizar esta sección, usted deberá ser capaz de realizar lo siguiente:

- Implementar una aplicación en un clúster desde un Dockerfile con la CLI.
- Describir los recursos creados en un proyecto con el comando `oc new-app` y la consola web.

Rutas de desarrollo

Red Hat OpenShift Container Platform está diseñada para compilar e implementar aplicaciones contenedorizadas. OpenShift admite dos casos de uso principales:

- Cuando el ciclo de vida completo de la aplicación, del desarrollo inicial a la producción, se administra con herramientas OpenShift.
- Cuando las aplicaciones contenedorizadas existentes, compiladas fuera de OpenShift, se implementan en 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 DO288 no abarca el recurso Deployment, solo DeploymentConfigs. Para crear recursos DeploymentConfig, puede pasar el indicador `--as-deployment-config` al invocar `oc new-app`. Usará el indicador `--as-deployment-config` en esta versión del curso. Para obtener más información, consulte Comprendión de Deployments y DeploymentConfigs [<https://docs.openshift.com/container-platform/4.5/applications/deployments/what-deployments-are.html#what-deployments-are>].

El comando `oc new-app` crea los recursos requeridos para compilar e implementar una aplicación en OpenShift. Se crean diferentes recursos, según el caso de uso deseado:

- Si desea que OpenShift administre el ciclo de vida completo de la aplicación, use el comando `oc new-app` para crear una configuración de compilación para administrar el proceso de compilación que crea la imagen de contenedor de la aplicación. El comando `oc new-app` también crea una configuración de implementación para administrar el proceso de implementación que ejecuta la imagen de contenedor generada en el clúster OpenShift. En el siguiente ejemplo, está delegando al clúster OpenShift el ciclo de vida completo: clonación de un repositorio Git, compilación de una imagen de contenedor e implementación en un clúster OpenShift.

```
[user@host ~]$ oc new-app --as-deployment-config \
> https://github.com/RedHatTraining/DO288/tree/master/apps/apache-httdp
```

- Si tiene una aplicación contenedoraizada existente que desea implementar en OpenShift, use el comando `oc new-app` para crear una configuración de implementación para administrar el proceso de implementación que ejecuta la imagen de contenedor existente en el clúster

OpenShift. En el siguiente ejemplo, se está haciendo referencia a una imagen de contenedor con la opción `--docker-image`:

```
[user@host ~]$ oc new-app --as-deployment-config \
> --docker-image=registry.access.redhat.com/rhel7-mysql57
```

El comando `oc new-app` también crea algunos recursos auxiliares, como servicios y flujos de imágenes. Estos recursos se requieren para admitir la manera en que OpenShift administra aplicaciones contenedorizadas y se presentan más adelante en este curso.

El botón **Add to Project** (Aregar a proyecto) de la consola web realiza las mismas tareas que el comando `oc new-app`. Un capítulo posterior de este curso abarca la consola web de OpenShift y su uso.

Descripción de las opciones del comando `oc new-app`

El comando `oc new-app` toma, en su forma más sencilla, un argumento de URL simple que apunta a un repositorio Git o una imagen de contenedor. Accede a la URL para determinar cómo interpretar el argumento y realizar una compilación o implementación.

Es posible que el comando `oc new-app` no pueda tomar la decisión que desea. Por ejemplo:

- Si un repositorio Git contiene un Dockerfile y un archivo `index.php`, OpenShift no puede identificar el método que se debe tomar, a menos que se mencione explícitamente.
- Si un repositorio Git contiene código fuente orientado a PHP, pero el clúster OpenShift admite la implementación de la versión 5.6 o 7.0 de PHP, el proceso de compilación falla porque no está claro qué versión usar.

Para dar cabida a estos y otros escenarios, el comando `oc new-app` proporciona una serie de opciones para especificar con mayor precisión cómo compilar la aplicación:

Opciones soportadas

| Opción | Descripción |
|-------------------------------------|--|
| <code>--as-deployment-config</code> | Configura el comando <code>oc new-app</code> para crear un recurso DeploymentConfig en lugar de un Deployment. |
| <code>--image-stream -i</code> | Proporciona el flujo de imágenes que se usará como la imagen de compilador S2I para una compilación S2I o para implementar una imagen de contenedor. |
| <code>--strategy</code> | <code>docker</code> o <code>pipeline</code> o <code>source</code> |
| <code>--code</code> | Proporciona la URL a un repositorio Git que se usará como entrada para una compilación S2I. |
| <code>--docker-image</code> | Proporciona la URL a una imagen de contenedor que se implementará. |

Administración del ciclo de vida completo de la aplicación con OpenShift

OpenShift administra el ciclo de vida de la aplicación mediante el proceso *Fuente a imagen (S2I)*. S2I toma el código fuente de la aplicación de un repositorio Git, lo combina con una imagen de

contenedor base, compila la fuente y crea una imagen de contenedor con la aplicación lista para su ejecución.

El comando `oc new-app` toma la URL de un repositorio Git como el argumento de entrada e inspecciona el código fuente de la aplicación para determinar la imagen de compilador que se usará para crear la imagen de contenedor de la aplicación:

```
[user@host ~]$ oc new-app --as-deployment-config http://gitserver.example.com/
mygitrepo
```

El comando `oc new-app` puede tomar opcionalmente el nombre del flujo de imágenes del compilador como un argumento, como parte de la URL de Git, antepuesto por una tilde (~) o mediante el argumento `--image-stream` (forma corta: `-i`).

Los siguientes dos comandos ilustran el uso de una imagen de compilador S2I de PHP:

```
[user@host ~]$ oc new-app --as-deployment-config php-http://gitserver.example.com/
mygitrepo
```

```
[user@host ~]$ oc new-app --as-deployment-config -i php http://
gitserver.example.com/mygitrepo
```

Opcionalmente, siga el nombre del flujo de imágenes acompañado por una etiqueta, que por lo general es el número de versión del tiempo de ejecución del lenguaje de programación. Por ejemplo:

```
[user@host ~]$ oc new-app --as-deployment-config php:7.0~http://
gitserver.example.com/mygitrepo
```

```
[user@host ~]$ oc new-app --as-deployment-config -i php:7.0 http://
gitserver.example.com/mygitrepo
```

Especificación del nombre del flujo de imágenes

Algunos desarrolladores prefieren la opción `-i` a la tilde, debido a que el carácter de tilde no es muy legible, dependiendo de la fuente de la pantalla. Los siguientes tres comandos producen los mismos resultados:

```
[user@host ~]$ oc new-app --as-deployment-config \
> myis-http://gitserver.example.com/mygitrepo
```

```
[user@host ~]$ oc new-app --as-deployment-config \
> -i myis http://gitserver.example.com/mygitrepo
```

```
[user@host ~]$ oc new-app --as-deployment-config -i myis --strategy source \
> --code http://gitserver.example.com/mygitrepo
```

Aunque el comando `oc new-app` pretende ser una manera conveniente de ofrecer aplicaciones, los desarrolladores deben tener en cuenta que el comando intentará "adivinar" el idioma de origen del repositorio Git dado.

Del ejemplo anterior se desprende que, si *myis* no es uno de los flujos de imágenes S2I estándares proporcionados por OpenShift, solo el primer ejemplo funciona. La notación de tildes deshabilita la funcionalidad de detección de idiomas del comando `oc new-app`. Esto permite el uso de un flujo de imágenes que apunta a un compilador para un lenguaje de programación que el comando `oc new-app` desconoce.

La tilde (~) y las opciones `--image-stream` (-i) no funcionan de la misma manera; la opción -i requiere que el cliente Git se instale localmente, ya que la detección de idiomas debe clonar el repositorio para que pueda inspeccionar el proyecto y la notación tilde (~) no.

Implementación de aplicaciones contenedorizadas existentes en OpenShift

Si desarrolla una aplicación fuera de OpenShift y la imagen de contenedor de la aplicación está disponible en un registro de imágenes de contenedores al que el clúster OpenShift puede acceder, el comando `oc new-app` puede tomar la URL de la imagen de contenedor como argumento de entrada:

```
[user@host ~]$ oc new-app --as-deployment-config \
> registry.example.com/mycontainerimage
```

Tenga en cuenta que no hay una manera de saber, del comando anterior, si la URL se refiere a un repositorio Git o a una imagen de contenedor dentro de un registro. El comando `oc new-app` accede a la URL de entrada para resolver esta ambigüedad. OpenShift inspecciona el contenido de la URL y determina si es un código fuente o un registro de imágenes de contenedores. Para evitar esta ambigüedad, use las opciones `--code` o `--docker-image`. Por ejemplo:

```
[user@host ~]$ oc new-app --as-deployment-config \
> --code http://gitserver.example.com/mygitrepo
```

```
[user@host ~]$ oc new-app --as-deployment-config \
> --docker-image registry.example.com/mycontainerimage
```

Implementación de Dockerfiles existentes con OpenShift

En muchos casos, tiene imágenes de contenedores existentes compiladas con Dockerfiles. Si se puede acceder a los Dockerfile desde un repositorio Git, el comando `oc new-app` puede crear una configuración de compilación que realiza la compilación del Dockerfile dentro del clúster OpenShift y, luego, envía la imagen de contenedor resultante al registro interno:

```
[user@host ~]$ oc new-app --as-deployment-config \
> http://gitserver.example.com/mydockerfileproject
```

OpenShift accede a la URL fuente para determinar si contiene un Dockerfile. Si el mismo proyecto contiene archivos fuente para lenguajes de programación, OpenShift puede crear una configuración de compilador para una compilación S2I en lugar de una compilación de Dockerfile. Para evitar la ambigüedad, use la opción `--strategy`:

```
[user@host ~]$ oc new-app --as-deployment-config \
> --strategy docker \
> http://gitserver.example.com/mydockerfileproject
```

En el siguiente ejemplo, se ilustra el uso de la opción `--strategy` para una compilación de S2I:

```
[user@host ~]$ oc new-app --as-deployment-config \
> --strategy source \
> http://gitserver.example.com/user/mygitrepo
```

Otras opciones, como `--image-stream` y `--code`, pueden usarse en el mismo comando con `--strategy`.



nota

El comando `oc new-app` también proporciona algunas opciones para crear aplicaciones creadas a partir de una plantilla o aplicaciones creadas por una tubería de Jenkins, pero no las cubriremos en este capítulo.

Recursos creados por el comando `oc new-app`

El comando `oc new-app --as-deployment-config` agrega los siguientes recursos al proyecto actual para admitir la compilación e implementación de una aplicación:

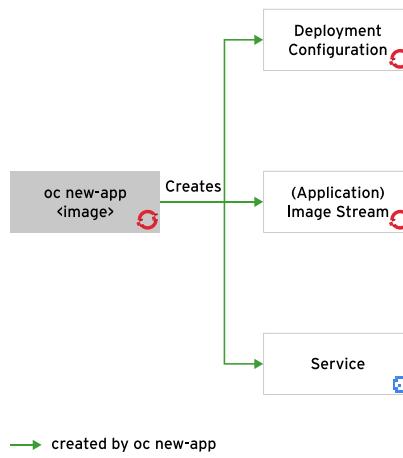
- Una configuración de compilación para compilar la imagen de contenedor de la aplicación a partir de código fuente o un Dockerfile.
- Un flujo de imágenes que apunta a la imagen generada en el registro interno o a una imagen existente en un registro externo.
- Una configuración de implementación que usa el flujo de imágenes como entrada para crear pods de aplicación.
- Un servicio para todos los puertos que expone la imagen de contenedor de la aplicación. Si la imagen de contenedor de la aplicación no declara puertos expuestos, el comando `oc new-app` no crea un servicio.

Estos recursos inician una serie de procesos que, a su vez, crean más recursos en el proyecto, como pods de aplicación para ejecutar aplicaciones contenedezadas.

El siguiente comando crea una aplicación basada en la imagen `mysql` con la etiqueta establecida como `db=mysql`:

```
[user@host ~]$ oc new-app --as-deployment-config \
> mysql 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 1.9: Recursos creados por el comando oc new-app --as-deployment-config**

El siguiente comando crea una aplicación del código fuente en el lenguaje de programación PHP:

```
[user@host ~]$ oc new-app --as-deployment-config \
> --name hello -i php \
> --code http://gitserver.example.com/mygitrepo
```

Después de que se completan los procesos de compilación e implementación, use el comando `oc get all` para mostrar todos los recursos en el proyecto `test`. En la salida, se muestran algunos recursos, además de los que se crearon con el comando `oc new-app`:

| NAME | TYPE | FROM | LATEST | | | |
|------------------|--------------|---|----------|----------------------------|---------|--|
| bc/hello | Source | Git | 3 ① | | | |
| builds/hello-1 | Source | Git@3a0af02 | Complete | About an hour ago | 1m16s ② | |
| is/hello | DOCKER REPO | docker-registry.default.svc:5000/test/hello | ③ | TAGS | UPDATED | |
| dc/hello | REVISION | DESIRED | CURRENT | TRIGGERED BY | | |
| dc/hello | 1 | 1 | 1 | config,image(hello:latest) | ④ | |
| rc/hello-1 | DESIRED | CURRENT | READY | AGE | | |
| rc/hello-1 | 1 | 1 | 1 | 3m ⑤ | | |
| svc/hello | CLUSTER-IP | EXTERNAL-IP | PORT(S) | AGE | | |
| svc/hello | 172.30.2.186 | <none> | 8080/TCP | 2m31s ⑥ | | |
| po/hello-1-build | READY | STATUS | RESTARTS | AGE | | |
| po/hello-1-tmf1 | 0/1 | Completed | 0 | 2m11s ⑦ | | |
| po/hello-1-tmf1 | 1/1 | Running | 0 | 1m23s ⑧ | | |

- ① La configuración de compilación creada por el comando `oc new-app`.
- ② La primera compilación se desencadena con el comando `oc new-app`.
- ③ El flujo de imágenes creado por el comando `oc new-app`. Apunta a la imagen de contenedor creada por el proceso S2I.

- ④ La configuración de implementación creada por el comando `oc new-app`.
- ⑤ La configuración del controlador de replicación creada por la primera implementación. Las implementaciones posteriores también podrían crear pods de implementador.
- ⑥ El servicio creado por el comando `oc new-app` como resultado de la imagen de compilador S2I de PHP que expone el puerto 8080/TCP.
- ⑦ OpenShift conserva los pods de compilación de las compilaciones más recientes debido a que usted podría querer inspeccionar esos registros. Los pods del implementador se eliminan después de la finalización correcta.
- ⑧ El pod de la aplicación creado por la primera implementación.

Una aplicación puede esperar una cantidad de recursos que no son creados por el comando `oc new-app`, como rutas, secretos y reclamaciones de volúmenes persistentes. Estos recursos pueden crearse con otros comandos `oc` antes o después de usar el comando `oc new-app`.

Todos los recursos creados por el comando `oc new-app` incluyen la etiqueta `app`. El valor de la etiqueta `app` coincide con el nombre corto del repositorio Git de la aplicación o la imagen de contenedor existente. Para especificar un valor diferente para la etiqueta `app`, use la opción `--name`; por ejemplo:

```
[user@host ~]$ oc new-app --as-deployment-config \
> --name test http://gitserver.example.com/mygitrepo
```

Puede eliminar recursos creados por el comando `oc new-app` con un único comando `oc delete` y la etiqueta `app`, sin tener que eliminar todo el proyecto y sin afectar otros recursos que puedan existir en el proyecto. El siguiente comando elimina todos los recursos creados por el comando `oc new-app` anterior:

```
[user@host ~]$ oc delete all -l app=test
```

Use el argumento de la opción `--name` para especificar el nombre base para los recursos creados por el comando `oc new-app`, como servicios y configuraciones de compilación.

El comando `oc new-app` se puede ejecutar varias veces dentro del mismo proyecto de OpenShift para crear aplicaciones de varios contenedores, una a la vez. Por ejemplo:

- Ejecute el comando `oc new-app` con la URL a una imagen de contenedor de base de datos MongoDB para crear un pod de base de datos y un servicio.
- Ejecute el comando `oc new-app` con la URL al repositorio Git para una aplicación Node.js que requiera acceso a la base de datos, usando el servicio creado por la primera invocación.

A continuación, puede exportar todos los recursos creados por ambos comandos en un archivo de plantilla.

Si desea inspeccionar las definiciones de los recursos sin crear los recursos en el proyecto actual, use la opción `-o`:

```
[user@host ~]$ oc new-app --as-deployment-config \
> -o json registry.example.com/mycontainerimage
```

Las definiciones de los recursos se envían a la salida estándar y se pueden redireccionar a un archivo. El archivo resultante se puede personalizar o insertar en una definición de plantilla.

**nota**

OpenShift proporciona varias plantillas predefinidas para escenarios comunes, como una base de datos más una aplicación. Por ejemplo, la plantilla `rails-postgresql` implementa una imagen de contenedor de base de datos PostgreSQL y una aplicación Ruby on Rails compilada de la fuente.

Para obtener una lista completa de las opciones soportadas por el comando `oc new-app`, y para ver una lista de ejemplos, ejecute el comando `oc new-app -h`.

Referencia a imágenes de contenedores mediante flujos de imágenes y etiquetas

La comunidad de OpenShift recomienda usar los recursos de *flujo de imágenes* para hacer referencia a las imágenes de contenedores en lugar de usar referencias directas a las imágenes de contenedores. Un recurso de flujo de imágenes apunta a una imagen de contenedor en el registro interno o en un registro externo, y almacena metadatos, como las etiquetas disponibles y sumas de comprobación del contenido de la imagen.

Tener metadatos de la imagen de contenedor en un flujo de imágenes permite a OpenShift realizar operaciones (como imágenes en caché) basadas en estos datos en lugar de acudir a un servidor de registro cada vez. También permite usar estrategias de notificación o agrupamiento para reaccionar a las actualizaciones del contenido de la imagen.

Las configuraciones de compilación y de implementación usan eventos de flujo de imágenes para realizar operaciones como las siguientes:

- Desencadenar una nueva compilación S2I debido a que se actualizó la imagen del compilador.
- Desencadenar una nueva implementación de pods para una aplicación debido a que se actualizó la imagen de contenedor de la aplicación en un registro externo.

La manera más sencilla de crear un flujo de imágenes es usar el comando `oc import-image` con la opción `--confirm`. En el siguiente ejemplo, se crea un flujo de imágenes denominado `myis` para la imagen de contenedor `acme/awesome` que proviene del registro inseguro en `registry.acme.example.com`:

```
[user@host ~]$ oc import-image myis --confirm \
> --from registry.acme.example.com:5000/acme/awesome --insecure
```

En el proyecto `openshift`, se proporcionan varios flujos de imágenes para el beneficio de todos los usuarios del clúster OpenShift. Puede crear sus propios flujos de imágenes en el proyecto actual mediante el uso del comando `oc new-app`, así como el uso de plantillas de OpenShift.

Un recurso de flujo de imágenes puede definir varias *etiquetas de flujo de imágenes*. Una etiqueta de flujo de imágenes puede apuntar a la etiqueta de una imagen de contenedor diferente o al nombre de una imagen de contenedor diferente. Esto significa que puede usar nombres más cortos y simples para imágenes comunes, como imágenes de compilador S2I, y usar nombres o registros diferentes para las variaciones de la misma imagen. Por ejemplo, el flujo de imágenes `ruby` del proyecto `openshift` define las siguientes etiquetas de flujo de imágenes:

- `ruby:2.5` hace referencia a `rhel8/ruby-25` de Red Hat Container Catalog.
- `ruby:2.6` hace referencia a `rhel8/ruby-26` de Red Hat Container Catalog.



Referencias

Puede encontrar más información en el capítulo *Comandos CLI del desarrollador de la referencia de CLI para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/cli_tools/index#cli-developer-commands

► Ejercicio Guiado

Implementación de una aplicación en un clúster OpenShift

En este ejercicio, usará OpenShift para compilar e implementar una aplicación a partir de un Dockerfile.

Resultados

Deberá ser capaz de crear una aplicación con la estrategia de compilación Docker y eliminar todos los recursos de la aplicación sin eliminar el proyecto.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen principal de la aplicación de muestra (ubi8/ubi).
- La aplicación de muestra en el repositorio Git (ubi-echo).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

```
[student@workstation ~]$ lab docker-build start
```

► 1. Inspeccione el Dockerfile para obtener la aplicación de muestra.

- 1.1. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación `master` del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 1.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b docker-build
Switched to a new branch 'docker-build'
[student@workstation D0288-apps]$ git push -u origin docker-build
...output omitted...
 * [new branch]      docker-build -> docker-build
Branch docker-build set up to track remote branch docker-build from origin.
```

- 1.3. Revise el Dockerfile para la aplicación, en la carpeta `ubi-echo`:

```
[student@workstation D0288-apps]$ cat ubi-echo/Dockerfile
FROM registry.access.redhat.com/ubi8/ubi:8.0 ①
USER 1001 ②
CMD bash -c "while true; do echo test; sleep 5; done" ③
```

- ① La imagen principal es la imagen base universal (UBI) para Red Hat Enterprise Linux 8.0 de Red Hat Container Catalog.
- ② El ID de usuario con el que se ejecuta esta imagen de contenedor. Cualquier valor distinto de cero funcionaría aquí. Solo para que sea diferente a los usuarios estándares del sistema, como apache, que generalmente se encuentran en el rango inferior de valores de UID.
- ③ La aplicación ejecuta un bucle que repite "test" cada cinco segundos.

► 2. Compile la imagen de contenedor de la aplicación con el clúster OpenShift.

- 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 D0288-apps]$ source /usr/local/etc/ocp4.config
```

- 2.2. Inicie sesión en OpenShift con su nombre de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

- 2.3. Cree un nuevo proyecto para la aplicación. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador.

```
[student@workstation D0288-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-docker-build
Now using project "youruser-docker-build" on server "https://
api.cluster.domain.example.com:6443".
```

- 2.4. Cree una nueva aplicación denominada "echo" desde el Dockerfile en la carpeta `ubi-echo`. Use la bifurcación que creó en el paso anterior. Crea, entre otros recursos, una configuración de compilación:

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config --name echo \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#docker-build \
> --context-dir ubi-echo
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "ubi" created
imagestream.image.openshift.io "echo" created
buildconfig.build.openshift.io "echo" created
deploymentconfig.apps.openshift.io "echo" created
--> Success
...output omitted...
```

Ignore las advertencias acerca de la imagen base que se ejecuta como raíz. Recuerde que su Dockerfile cambió a un usuario sin privilegios.

2.5. Siga los registros de compilación:

```
[student@workstation D0288-apps]$ oc logs -f bc/echo
Cloning "https://github.com/youruser/D0288-apps#docker-build" ...
Replaced Dockerfile FROM image registry.access.redhat.com/ubi8/ubi:8.0
Caching blobs under "/var/cache/blobs".

...output omitted...
Pulling image registry.access.redhat.com/ubi8/ubi@sha256:1a2a...75b5
...output omitted...
STEP 1: FROM registry.access.redhat.com/ubi8/ubi@sha256:1a2a...75b5 ①
STEP 2: USER 1001
STEP 3: CMD bash -c "while true; do echo test; sleep 5; done"
STEP 4: ENV "OPENSHIFT_BUILD_NAME"="echo-1" ... ②
STEP 5: LABEL "io.openshift.build.commit.author"=...
STEP 6: COMMIT containers-storage:[overlay@/var/lib/containers/storage+... ③
...output omitted...
Pushing image image-registry.openshift-image-registry.svc:5000/youruser-docker-
build/echo:latest ... ④
Push successful
```

- ① El comando `oc new-app` identificó correctamente el repositorio Git como un proyecto Dockerfile y la compilación de OpenShift realiza una compilación de Dockerfile.
- ② OpenShift agrega metadatos a la imagen de contenedor de la aplicación con las instrucciones `ENV` y `LABEL`.
- ③ OpenShift confirma la imagen de la aplicación en el motor del contenedor del nodo.
- ④ OpenShift envía la imagen de la aplicación del motor de contenedor del nodo al registro interno del clúster.

► 3. Verifique que la aplicación funciona dentro de OpenShift.

3.1. Espere a que se implemente la imagen de contenedor de la aplicación. Repita el comando `oc status` hasta que la salida muestre una implementación exitosa:

```
[student@workstation D0288-apps]$ oc status
In project youruser-docker-build on server
https://api.cluster.domain.example.com:6443

dc/echo deploys istag/echo:latest <-
bc/echo docker builds https://github.com/youruser/D0288-apps#docker-build on
istag/ubi:8.0
deployment #1 deployed 6 minutes ago - 1 pod
...output omitted...
```

3.2. Espere a que el pod de la aplicación esté listo y en ejecución. Repita el comando `oc get pod` hasta que la salida sea similar a la siguiente:

```
[student@workstation D0288-apps]$ oc get pod
NAME        READY   STATUS    RESTARTS   AGE
echo-1-build 0/1     Completed  0          1m
echo-1-deploy 0/1     Completed  0          1m
echo-1-555xx  1/1     Running   0          14s
```

- 3.3. Consulte los registros del pod de la aplicación para ver si la imagen de contenedor de la aplicación está produciendo la salida esperada en OpenShift. Use el nombre del pod de la aplicación que obtuvo del paso anterior.

```
[student@workstation D0288-apps]$ oc logs echo-1-555xx | tail -n 3
test
test
test
```

- 4. Inspeccione la configuración de compilación y de implementación para ver cómo se relacionan con el flujo de imágenes.

- 4.1. Revise la configuración de compilación:

```
[student@workstation D0288-apps]$ oc describe bc echo
Name:           echo
...output omitted...
Labels:         app=echo
...output omitted...
Strategy:      Docker
URL:           https://github.com/youruser/D0288-apps
Ref:            docker-build ①
ContextDir:    ubi-echo ②
From Image:    ImageStreamTag ubi:8.0 ③
Output to:     ImageStreamTag echo:latest ④
...output omitted...
```

- ① Las compilaciones se inicián desde la bifurcación docker-build desde el repositorio Git en el atributo URL.
- ② Las compilaciones toman solo la carpeta ubi-echo del repositorio Git en el atributo URL.
- ③ Las compilaciones toman un flujo de imágenes que apunta a la imagen principal de Dockerfile para que las nuevas compilaciones se puedan desencadenar por los cambios de imagen.
- ④ Las compilaciones generan una nueva imagen de contenedor y la envían al registro interno a través de un flujo de imágenes.

- 4.2. Revise el flujo de imágenes:

```
[student@workstation D0288-apps]$ oc describe is echo
Name:           echo
...output omitted...
Labels:         app=echo
...output omitted...
Image Repository: image-registry.openshift-image-registry.svc:5000/youruser-
docker-build/echo ①
```

```
...output omitted...
latest
no spec tag

* image-registry.openshift-image-registry.svc:5000/youruser-docker-build/
echo@sha256:5bbf...ef0b ②
...output omitted...
```

- ① El flujo de imágenes apunta al registro interno de OpenShift mediante el nombre DNS del servicio.
- ② Un hash SHA256 identifica la última imagen. Usando este hash, el flujo de imágenes puede detectar si se cambió la imagen.

4.3. Revise la configuración de implementación:

```
[student@workstation D0288-apps]$ oc describe dc echo
Name:           echo
...output omitted...
Labels:         app=echo
Triggers:       Config, Image(echo@latest, auto=true) ①
...output omitted...
Pod Template:
...output omitted...
Containers:
echo:
  Image:      docker-registry.default.svc:5000/youruser-docker-build/
echo@sha256:5bbf...ef0b ②
...output omitted...
Deployment #1 (latest):
...output omitted...
```

- ① La configuración de implementación tiene un desencadenador en el flujo de imágenes. Si el flujo de imágenes cambia, se realiza una nueva implementación.
- ② La plantilla del pod que está dentro de la configuración de implementación especifica el hash SHA256 de la imagen de contenedor para admitir estrategias de implementación como actualizaciones graduales.

► 5. Cambie la aplicación.

5.1. Edite la instrucción CMD en el Dockerfile en ~/D0288-apps/ubi-echo/Dockerfile para mostrar un contador. El contenido final de Dockerfile debe ser el siguiente:

```
FROM registry.access.redhat.com/ubi8/ubi:8.0
USER 1001
CMD bash -c "while true; do (( i++ )); echo test \$i; sleep 5; done"
```

5.2. Confirme y envíe los cambios al servidor Git.

```
[student@workstation D0288-apps]$ cd ubi-echo
[student@workstation ubi-echo]$ git commit -a -m 'Add a counter'
...output omitted...
[student@workstation ubi-echo]$ git push
...output omitted...
[student@workstation ubi-echo]$ cd ~
[student@workstation ~]$
```

- 6. Vuelva a compilar la aplicación y verifique que OpenShift implemente la nueva imagen de contenedor.

6.1. Inicie una nueva compilación de OpenShift:

```
[student@workstation ~]$ oc start-build echo
build.build.openshift.io/echo-2 started
```

6.2. Siga los nuevos registros de compilación para que finalice la compilación:

```
[student@workstation ~]$ oc logs -f bc/echo
...output omitted...
Push successful
```

6.3. Verifique que OpenShift inicie una nueva implementación después de que finaliza la compilación:

```
[student@workstation ~]$ oc status
...output omitted...
dc/echo deploys istag/echo:latest <-
  bc/echo docker builds https://github.com/youruser/D0288-apps#docker-build on
  istag/ubi:8.0
  deployment #2 deployed 50 seconds ago - 1 pod
  deployment #1 deployed 26 minutes ago
...output omitted...
```

6.4. Espere hasta que el nuevo pod de aplicación esté listo y en ejecución:

```
[student@workstation ~]$ oc get pod
NAME        READY   STATUS    RESTARTS   AGE
echo-1-build  0/1     Completed  0          27m
echo-1-deploy 0/1     Completed  0          27m
echo-2-build  0/1     Completed  0          1m
echo-2-deploy 0/1     Completed  0          1m
echo-2-p11hg  1/1     Running   0          1m
```

6.5. Consulte los registros del pod de la aplicación para ver si está ejecutando la nueva imagen de contenedor. Use el nombre del pod del paso anterior:

```
[student@workstation ~]$ oc logs echo-2-pl1hg | head -n 3
test 1
test 2
test 3
```

► 7. Compare el estado del flujo de imágenes antes y después de volver a compilar la aplicación.

Inspeccione el estado actual del flujo de imágenes:

```
[student@workstation ~]$ oc describe is echo
Name: echo
...output omitted...
Labels: app=echo
...output omitted...
latest
no spec tag

* image-registry.openshift-image-registry.svc:5000/youruser-docker-build/
echo@sha256:025a...542f ①
2 minutes ago
image-registry.openshift-image-registry.svc:5000/youruser-docker-build/
echo@sha256:5bbf...ef0b ②
...output omitted...
```

- ① Esta es la nueva imagen. Tenga en cuenta que el hash SHA256 es diferente al de la imagen anterior.
- ② Esta es la imagen anterior.

► 8. Elimine todos los recursos de la aplicación.

- 8.1. Use el comando `oc delete` con la etiqueta de la aplicación generada por el comando `oc new-app`:

```
[student@workstation ~]$ oc delete all -l app=echo
pod "echo-2-pl1hg" deleted
replicationcontroller "echo-1" deleted
replicationcontroller "echo-2" deleted
deploymentconfig.apps.openshift.io "echo" deleted
buildconfig.build.openshift.io "echo" deleted
build.build.openshift.io "echo-1" deleted
build.build.openshift.io "echo-2" deleted
imagestream.image.openshift.io "echo" deleted
imagestream.image.openshift.io "ubi" deleted
```

- 8.2. Verifique que no hayan quedado recursos en el proyecto. Debe eliminar todos los recursos de la aplicación en el proyecto. Si en la salida se muestra un pod en el estado **Terminating** (Finalizando), repita el comando hasta que desaparezca el pod.

```
[student@workstation ~]$ oc get all
No resources found.
```

Finalizar

En `workstation`, ejecute el comando `lab docker-build finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab docker-build finish
```

Esto concluye el ejercicio guiado.

Administración de aplicaciones con la consola web

Objetivos

Tras finalizar esta sección, usted deberá ser capaz de usar la consola web para lo siguiente:

- Implementar una aplicación desde una imagen binaria y administrar sus recursos.
- Ver los registros del pod y de compilación.
- Editar definiciones de recursos.

Descripción general de la consola web de OpenShift

La consola web de OpenShift es una interfaz de usuario basada en navegador que proporciona una alternativa gráfica a las tareas más comunes requeridas para administrar proyectos y aplicaciones de OpenShift. La funcionalidad que proporciona la consola web se centra principalmente en las tareas y el flujo de trabajo del desarrollador. La consola web no proporciona funcionalidad completa de administración de clústeres. Esta funcionalidad generalmente requiere el uso del comando oc.

Para acceder a la consola web, use la URL de la API de OpenShift, que para los clústeres principales es, por lo general, una URL HTTPS al nombre del host público principal.

En la página de inicio de la consola web, se muestra una lista de los proyectos a los que puede acceder el usuario actual. Desde la página de inicio, puede crear nuevos proyectos, eliminar existentes y dirigirse a la página de descripción general de un proyecto.

| Name | Display Name | Status | Requester |
|--------------|-----------------|--------|-----------|
| your-project | No display name | Active | youruser |

Figura 1.10: Lista de proyectos de la consola web

Se espera que pase la mayor parte de su tiempo usando páginas de descripción general de proyectos y las diferentes páginas de recursos de proyectos. En la página de descripción general del proyecto, se muestran datos de resumen acerca de las aplicaciones incluidas en el proyecto y

el estado de todos los pods de la aplicación. Desde la página de descripción general del proyecto, puede ir a las páginas de detalles de los recursos y agregar aplicaciones al proyecto.

Figura 1.11: Descripción general del proyecto de la consola web

Aplicaciones en OpenShift

La consola web de OpenShift define una aplicación como un conjunto de recursos que tienen el mismo valor para la etiqueta app. El comando `oc new-app` agrega esta etiqueta a todos los recursos de la aplicación que crea. La sección **+Add (+Aregar)** en la perspectiva de **Developer** (Desarrollador) brinda características similares al comando `oc new-app`, incluida la adición de la etiqueta app a los recursos.

Figura 1.12: Acciones disponibles desde la sección +Add (+Aregar)

La sección **Add (Agregar)** es el punto de entrada a un asistente que le permite elegir entre imágenes de compilador S2I, plantillas e imágenes de contenedor para implementar una

aplicación en el clúster de OpenShift como parte de un proyecto específico. El asistente clasifica las imágenes y plantillas en el catálogo según las etiquetas en las imágenes de contenedor y las anotaciones en las plantillas y los recursos del flujo de imágenes.

Páginas de detalles del recurso

La mayoría de las páginas de detalles de recursos enumeran todos los recursos del proyecto de un determinado tipo. Proporcionan enlaces para eliminar recursos específicos y acceder a la página de detalles de cada recurso.

En la página de detalles de un único recurso, se brinda información de estado personalizada para cada tipo de recurso en diferentes pestañas. Por ejemplo:

- En la página de detalles de una compilación, se muestran el historial, la configuración, el entorno y los registros de cada compilación.
- En la página de detalles de una implementación, se muestran el historial, la configuración, el entorno, los eventos y los registros de cada compilación.
- En la página de detalles de un servicio, se muestran el conjunto de pods con balanceo de carga por servicio y las rutas (si corresponde) que apuntan al servicio.
- En la página de detalles de un pod, se muestran el estado, la configuración, el entorno, los registros y los eventos para cada pod. También permite abrir una sesión de terminal que ejecute una shell dentro de cualquier contenedor del pod.

Por lo general, en la página de detalles de recursos, se enumeran todas las etiquetas asociadas con ese recurso. OpenShift usa etiquetas para registrar relaciones entre recursos. Por ejemplo, todos los pods creados por una configuración de implementación tienen la etiqueta `deploymentconfig` con el nombre de la configuración de implementación.

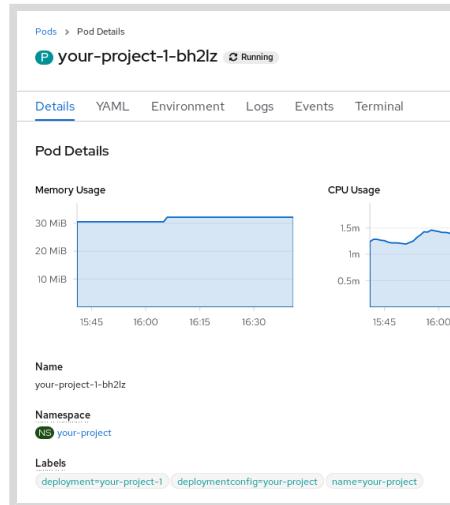


Figura 1.13: Etiquetas ubicadas en la parte inferior de la página de detalles de un pod

Por lo general, cuando en la consola web se muestra el nombre de un recurso, se trata de un enlace a la página de detalles de ese recurso. En la barra de navegación ubicada en la parte izquierda de la consola web, se brinda acceso a la página de detalles para todos los tipos de recursos soportados. En la parte superior de la barra de navegación, un ícono de inicio brinda acceso a la página de la lista de proyectos.

Acceso a registros con la consola web

La página de detalles del pod en la consola web incluye la pestaña **Logs** (Registros), en la que se muestran los registros del pod. El tiempo de ejecución del contenedor recopila la salida estándar de los contenedores que están dentro de un pod y los almacena como registros de pod.



nota

Solo los registros escritos en la salida estándar dentro del contenedor se pueden ver con el comando `oc logs` o en la consola web. Si una aplicación contenedorizada guarda los eventos de su registro en archivos de registro, ya sea en un almacenamiento de contenedor efímero o en un volumen persistente, OpenShift no muestra los registros en la consola web, ni con el comando `oc logs`.

La pestaña **Logs** (Registros) se actualiza automáticamente con las últimas entradas de registro. Esta pestaña también proporciona las siguientes acciones:

- Use el enlace **Download** (Descargar) para descargar y guardar los registros en un archivo local.
- Use el enlace **Expand** (Expandir) para hacer que los registros de pod consuman toda la pantalla para facilitar la visualización.

Las compilaciones e implementaciones son operaciones realizadas por OpenShift mediante pods de compilador e implementador. En la página **Build Details** (Detalles de compilación), se capturan y se almacenan los registros del pod de compilador. Use esta página para ver estos registros del pod de compilador. OpenShift no almacena los registros de una implementación, a menos que se produjeran errores.

```

Builds > Build Details
B your-project-1 Complete Actions ▾

Details YAML Environment Logs Events

Log stream ended.
56 lines
Copying config sha256:c8fb8cb622514af059a359153cbf76b7db3270816d4b7ef5a48b2a0080f1dbb0
Writing manifest to image destination
Storing signatures
--> c8fb8cb6225
c8fb8cb622514af059a359153cbf76b7db3270816d4b7ef5a48b2a0080f1dbb0

Pushing image image-registry.openshift-image-registry.svc:5000/your-project/your-project:latest ...
Getting image source signatures
Copying blob sha256:02cf8840727e51a3c232637b7496e15a9a5671885a64ff0dse4a9dc2c1674fd
Copying blob sha256:9e7a6dc796f0a75c560158a9f9e30fb85ba90cb53edce9ffbd5778406e4de39
Copying blob sha256:fc5b206e9329a1674dd9e8efbee45c9be280d0dcbabba3c6bb67a2f2cfcf2a
Copying blob sha256:e7021e0589e97471d99c4265b7c8e64da328e4f116bf260353b2e0a2ad373
Copying blob sha256:9a0a629e11c896fb5c92c3ede513fa50843042f978963a2a9e31288d1a7d5489
Copying config sha256:c8fb8cb622514af059a359153cbf76b7db3270816d4b7ef5a48b2a0080f1dbb0
Writing manifest to image destination
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/your-project/your-project@sha256:b3c84caa0cadadaa4bd2a3a502e7b2fa12b
Push successful

```

Figura 1.14: La pestaña **Logs** (Registros) de la página **Build Details** (Detalles de compilación)

Administración de compilaciones e implementaciones con la consola web

En la consola web de OpenShift, se proporcionan características para administrar configuraciones de compilación y de implementación, así como todas las compilaciones e implementaciones individuales desencadenadas por cada configuración. Gran parte de esta configuración se

realiza en las páginas de detalles que son específicas de la configuración de compilación o de la configuración de implementación que desea actualizar.

En la página de detalles de una configuración de compilación, se proporcionan las pestañas **Details** (Detalles), **YAML**, **Builds** (Compilaciones), **Environment** (Entorno) y **Events** (Eventos), así como el botón **Actions** (Acciones), que tiene la acción **Start Build** (Iniciar compilación). Esta acción realiza la misma función que el comando `oc start-build`.

Si el repositorio del código fuente de la aplicación no está configurado para usar los webhooks de OpenShift, necesita usar la consola web o la CLI para desencadenar nuevas compilaciones, después de enviar actualizaciones al código fuente de la aplicación.

En la página de detalles de la configuración de implementación, se proporcionan muchas más funcionalidades, incluidas acciones para personalizar diversos aspectos de una configuración de implementación, como recuento o almacenamiento de pods deseados.

En la página de detalles de una implementación, también se proporciona el botón **Action** (Acción) con diferentes acciones de la página de detalles de configuración de la compilación. La acción **Start Rollout** (Iniciar lanzamiento) está disponible y realiza la misma función que el comando `oc rollout latest`. Necesita usar la CLI para realizar operaciones `oc rollout` más específicas.

Edición de recursos de OpenShift

En la mayoría de las páginas de detalles de recursos de la consola web de OpenShift, se proporciona un botón **Actions** (Acciones) que muestra un menú. Este menú puede ofrecer algunas de las siguientes opciones:

- **Edit resource** (Editar recurso): Edite un recurso existente mediante la sintaxis YAML sin formato, en un editor de texto basado en navegador con resultado de sintaxis. Esta acción es equivalente a usar el comando `oc edit -o yaml`.
- **Delete resource** (Eliminar recurso): Elimine un recurso existente. Esta acción es equivalente a usar el comando `oc delete`.
- **Edit Labels** (Editar etiquetas): abre un cuadro de diálogo modal para editar las etiquetas de recursos.
- **Edit Annotations** (Editar anotaciones): abre un cuadro de diálogo modal para editar las claves y los valores de las anotaciones.

No todas las operaciones de administración de recursos pueden realizarse a través de la consola web. Las operaciones del administrador de clústeres, en particular, normalmente requieren la CLI.



Referencias

Para obtener más información acerca de la organización, la navegación y el uso de la consola web, consulte la guía *Consola web para Red Hat OpenShift Container Platform 4.5* en
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/web_console/index

► Ejercicio Guiado

Administración de una aplicación con la consola web

En este ejercicio, usará la consola web de OpenShift para implementar una imagen de contenedor del servidor HTTP de Apache.

Resultados

Deberá ser capaz de usar la consola web de OpenShift para lo siguiente:

- Crear un nuevo proyecto y agregar una nueva aplicación para implementar una imagen de contenedor.
- Realizar tareas comunes de solución de problemas, como ver registros, inspeccionar definiciones de recursos y eliminar recursos.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen de contenedor para la aplicación de muestra (`redhattraining/php-hello-dockerfile`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos:

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

- 1. Abra un navegador web y diríjase a `https://console-openshift-console.apps.cluster.domain.example.com` para acceder a la consola web de OpenShift. Inicie sesión y cree un nuevo proyecto con el nombre `youruser-deploy-image`.
- 1.1. Busque el dominio comodín del clúster OpenShift. Es la variable `RHT_OCP4_WILDCARD_DOMAIN` en el archivo `/usr/local/etc/ocp4.config` de configuración del aula.

```
[student@workstation ~]$ grep RHT_OCP4_WILDCARD_DOMAIN /usr/local/etc/ocp4.config
RHT_OCP4_WILDCARD_DOMAIN=apps.cluster.domain.example.com
```

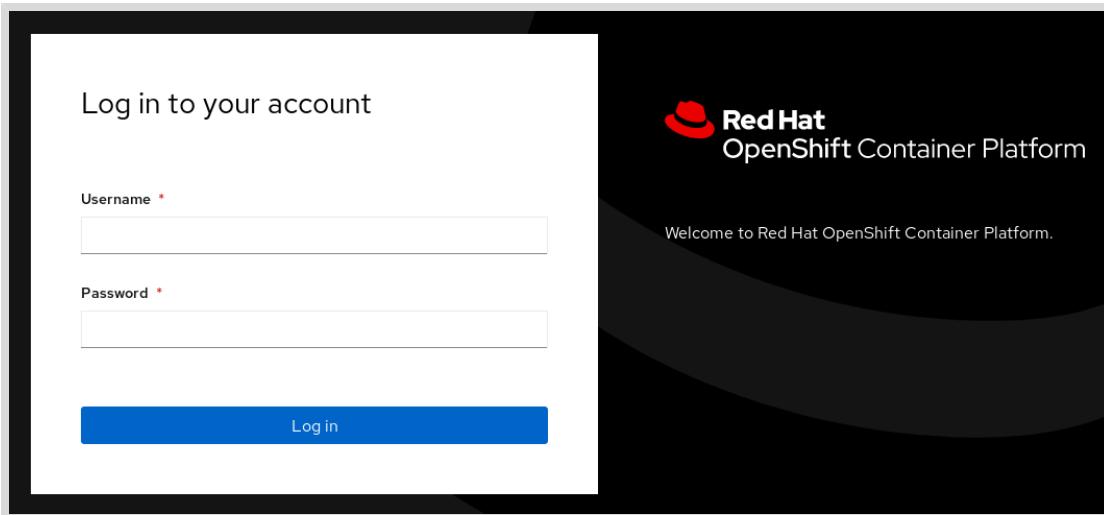
Otra forma de encontrar el nombre de host de su consola web de OpenShift es inspeccionar las rutas en el proyecto `openshift-console`. Debe haber iniciado sesión en OpenShift antes.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
[student@workstation ~]$ oc get route -n openshift-console
NAME          HOST/PORT          PATH ...
console      console-openshift-console.apps.cluster.domain.example.com ...
downloads    downloads-openshift-console.apps.cluster.domain.example.com ...
```

**nota**

La configuración predeterminada de Red Hat OpenShift Container Platform no permite a los usuarios regulares acceder al proyecto `openshift-console`, pero el entorno del aula se configuró para otorgar estos permisos.

- 1.2. Abra un navegador web y diríjase a `https://console-openshift-console.apps.cluster.domain.example.com` para acceder a la consola web de OpenShift. Reemplace `apps.cluster.domain.example.com` con el valor que obtuvo del paso anterior. Debería ver la página de inicio de sesión para la consola web.
- 1.3. Inicie sesión con el usuario de desarrollador. Su nombre de usuario (`youruser`) es la variable `RHT_OCP4_DEV_USER` en el archivo de configuración del aula `/usr/local/etc/ocp4.config`. Su contraseña es el valor de la variable `RHT_OCP4_DEV_PASSWORD` en el mismo archivo.



- 1.4. Diríjase a la perspectiva de administrador y, luego, seleccione **Home (Inicio) → Projects (Proyectos)** en el menú de la izquierda. Los usuarios nuevos se colocan en esa página inicialmente, haga clic en **Create Project** (Crear proyecto). En el cuadro de diálogo **Create Project** (Crear proyecto), ingrese `youruser-deploy-image` en el campo **Name (Nombre)**. Reemplace `youruser` con el valor de su variable `RHT_OCP4_DEV_USER`. No necesita completar los campos de nombre de visualización y descripción.

Haga clic en **Create (Crear)** para crear el nuevo proyecto.

- 2. Cree una nueva aplicación desde una imagen de contenedor creada previamente que contenga una aplicación "Hello, World" escrita en PHP y cree una ruta para exponer la aplicación públicamente.
- 2.1. En la página **Project Details** (Detalles del proyecto), haga clic en la pestaña **Workloads** (Cargas de trabajo) y, luego, en el enlace **Add Other Content** (Agregar otro contenido).

- 2.2. En la página **Add** (Agregar), haga clic en el botón **Container Image** (Imagen de contenedor).

- 2.3. En la página **Deploy Image** (Implementar imagen), ingrese `quay.io/redhattraining/php-hello-dockerfile` en el campo **Image Name** (Nombre de la imagen). La consola web de OpenShift se conecta con el registro público `quay.io` y recupera información sobre la imagen del contenedor.

- 2.4. Desplácese hacia abajo para ver información acerca de la imagen y reemplace `php-hello-dockerfile` con `hello` en los campos **Name** (Nombre) y **Application Name** (Nombre de la aplicación).

Seleccione **Deployment Config** (Configuración de implementación) en la sección **Resources** (Recursos).

Desmarque la opción **Create a Route to the Application** (Crear una ruta a la aplicación) en la sección **Advanced Options** (Opciones avanzadas) en la parte inferior de la página.

- 2.5. Haga clic en **Create** (Crear) para crear la nueva aplicación. La consola web crea todos los recursos de OpenShift requeridos para implementar la imagen de contenedor y cambia a la página **Topology** (Topología) del proyecto.

- 2.6. Haga clic en el ícono sobre el nombre de implementación para expandir los detalles de la implementación y ver la cantidad de pods en ejecución. Espere hasta que en la página de descripción general se muestre una implementación exitosa con un pod:

2.7. En la barra de navegación, haga clic en **Networking (Redes)** → **Routes (Rutas)**.

En la página **Routes (Rutas)**, haga clic en el botón **Create Route (Crear ruta)**.

Ingrese **hello-route** en el campo **Name (Nombre)**. Ingrese **hello-youruser.apps.cluster.domain.example.com** en el campo **Hostname (Nombre de host)**.

Reemplace *youruser* con el valor de su variable RHT_OCP4_DEV_USER; es decir, el nombre de usuario que usó para iniciar sesión en OpenShift. Reemplace *apps.cluster.domain.example.com* con el valor de la variable RHT_OCP4_WILDCARD_DOMAIN.

Desplácese hacia abajo y seleccione el servicio **Hello** de la lista **Service (Servicio)**. En la lista **Target Port (Puerto de destino)**, seleccione **8080** → **8080 (TCP)**. No modifique los demás campos. Desplácese hacia abajo y haga clic en **Create (Crear)**.

The screenshot shows the Red Hat OpenShift Container Platform web interface. On the left, a sidebar menu is open under the 'Networking' section, specifically the 'Routes' sub-section. The main area is titled 'Project: youruser-deploy-image'. It contains fields for 'Name' (set to 'hello-route'), 'Hostname' (set to 'hello-youruser.apps.cluster.domain.example.com'), 'Path' (set to '/'), 'Service' (set to 'hello'), and 'Target Port' (set to '8080 + 8080 (TCP)'). Below these fields are 'Security' options, including a checkbox for 'Secure route'. At the bottom are 'Create' and 'Cancel' buttons.

- 2.8. La página de detalles de la ruta cambia y se muestra la URL de acceso a la aplicación, mediante la nueva ruta.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The sidebar menu is open under the 'Networking' section, specifically the 'Routes' sub-section. The main area displays the details of a route named 'hello-route'. The status is shown as 'Accepted'. The 'Details' tab is selected, showing the route's configuration: Name (hello-route), Location (http://hello-youruser.apps.cluster.domain.example.com), Status (Accepted), Host (hello-youruser.apps.cluster.domain.example.com), and Path (/). The 'YAML' tab is also visible.

Haga clic en <http://hello-youruser.apps.cluster.domain.example.com> para abrir una nueva pestaña del navegador que muestre la página predeterminada devuelta por la aplicación PHP. Es un simple mensaje "Hello, World" con la versión de PHP.

- 3. Explore las características de solución de problemas de la consola web.

- 3.1. Consulte los registros del pod de la aplicación.

En la barra de navegación, haga clic en **Workloads (Cargas de trabajo) → Pods**.

| Name | Namespace | Status | Ready | Owner | Memory | CPU |
|----------------|-----------------------|-----------|-------|------------|----------|-------------|
| hello-1-deploy | psolarvi-deploy-image | Completed | 0/1 | RC hello-1 | - | - |
| hello-1-rtdgd | psolarvi-deploy-image | Running | 1/1 | RC hello-1 | 77.2 MiB | 0.001 cores |

Haga clic en el nombre del pod de la aplicación, como **hello-1-bmrc9** para ver la página **Pod Details** (Detalles del pod). Haga clic en la pestaña **Logs** (Registros) para ver los registros del pod. Aparecerá el mensaje de advertencia acerca del nombre de dominio completamente calificado del servidor, que se puede ignorar de forma segura.

```

[04-Aug-2020 08:10:59] NOTICE: [pool www] 'user' directive is ignored when FPM is not running as root
[04-Aug-2020 08:10:59] NOTICE: [pool www] 'group' directive is ignored when FPM is not running as root
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 10.131.1.68. Set

```

3.2. Inicie una sesión de shell dentro de un contenedor en ejecución.

En la página **Pod Details** (Detalles del pod), haga clic en la pestaña **Terminal** para abrir una shell remota en el contenedor del pod de la aplicación. Si la ventana de terminal es demasiado pequeña, haga clic en **Expand** (Expandir) para ocultar los paneles de navegación de la consola web. El terminal solo puede ejecutar comandos que existan dentro de la imagen de contenedor de la aplicación. El comando `ps` no está disponible, pero puede ver los registros de acceso del servidor HTTP de Apache.

```

sh-4.4$ ps ax
sh: ps: command not found
sh-4.4$
sh-4.4$ cat /var/log/httpd/access_log
10.131.0.1 - - [04/Aug/2020:08:39:57 +0000] "GET / HTTP/1.1" 200 36 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
10.131.0.1 - - [04/Aug/2020:08:39:57 +0000] "GET /favicon.ico HTTP/1.1" 404 209 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0"
sh-4.4$
sh-4.4$ cat /var/log/php-fpm/error.log
[04-Aug-2020 08:10:59] NOTICE: [pool www] 'user' directive is ignored when FPM is not running as root
[04-Aug-2020 08:10:59] NOTICE: [pool www] 'group' directive is ignored when FPM is not running as root
[04-Aug-2020 08:10:59] NOTICE: fpm is running, pid 9
[04-Aug-2020 08:10:59] NOTICE: ready to handle connections
[04-Aug-2020 08:10:59] NOTICE: systemd monitor interval set to 10000ms
sh-4.4$ 

```

Si es necesario, haga clic en **Collapse** (Contraer) para volver a mostrar los paneles de navegación de la consola web.

3.3. Consulte la definición de un recurso.

También en la página **Pod Details** (Detalles del pod), haga clic en la pestaña **YAML**.

```

1 kind: Pod
2 apiVersion: v1
3 metadata:
4   generateName: hello-1-
5   annotations:
6     k8s.v1.cni.cncf.io/network-status: |
7       {
8         "name": "openshift-sdn",
9         "interface": "eth0",
10        "ips": [
11          "10.131.1.68"
12        ],
13        "default": true,
14        "dns": {}

```

La pestaña incluye un bonito editor web enriquecido para la sintaxis de YAML. No realice ningún cambio y haga clic en **Cancel** (Cancelar) para salir del editor.

► 4. Elimine los recursos del proyecto.

- 4.1. En la página **Pod Details** (Detalles del pod), haga clic en **Actions (Acciones)** → **Delete Pod (Eliminar pod)**. En el cuadro de diálogo de confirmación, haga clic en **Delete (Eliminar)** para eliminar el pod en ejecución. En la consola web, se muestra la página **Pods**.
Espere hasta que en la página **Pods** se muestre que la configuración de implementación creó un nuevo pod para reemplazar el pod eliminado.
- 4.2. En la barra de navegación, haga clic en **Workloads (Cargas de trabajo)** → **Deployment Configs (Configuraciones de implementación)** para ver la página **Deployment Configs (Configuraciones de implementación)**. Haga clic en **hello** para ver la página de detalles de la configuración de implementación.
En la esquina superior derecha, haga clic en **Actions (Acciones)** → **Delete Deployment Config (Eliminar configuración de implementación)**. En el cuadro de diálogo de confirmación, deje la casilla de verificación marcada y haga clic en **Delete (Eliminar)** para eliminar la configuración de implementación.
- 4.3. En la barra de navegación, haga clic en **Networking (Redes)** → **Services (Servicios)** y observe que aún hay un servicio en el proyecto.
Haga clic en **hello** para acceder a la página **Service Details** (Detalles del servicio). En la esquina superior derecha, haga clic en **Actions (Acciones)** → **Delete Service (Eliminar servicio)**. En el cuadro de diálogo de confirmación, haga clic en **Delete (Eliminar)** para eliminar el servicio.
- 4.4. En la barra de navegación, haga clic en **Networking (Redes)** → **Routes (Rutas)** y observe que aún hay una ruta en el proyecto.
Haga clic en **hello-route** para acceder a la página **Route Details** (Detalles de la ruta). En la esquina superior derecha, haga clic en **Actions (Acciones)** → **Delete Route (Eliminar ruta)**. En el cuadro de diálogo de confirmación, haga clic en **Delete (Eliminar)** para eliminar la ruta.

► **5.** Elimine el proyecto.

En la esquina superior izquierda, haga clic en **Home (Inicio)** → **Projects (Proyectos)** para ver la página **Projects (Proyectos)**. Haga clic en el ícono del menú a la derecha del proyecto **youruser-deploy-image** y, luego, haga clic en **Delete Project (Eliminar proyecto)**. Ingrese **youruser-deploy-image** en el cuadro de diálogo de confirmación y, luego, haga clic en **Delete (Eliminar)** para eliminar el proyecto.

Espere hasta que el proyecto **youruser-deploy-image** desaparezca de la página **Projects (Proyectos)**. Recuerde que no necesita eliminar los recursos de la aplicación uno por uno, como hizo en el paso anterior. Si elimina un proyecto, se eliminan todos los recursos dentro del proyecto.

Finalizar

En **workstation**, ejecute el comando **lab deploy-image finish** para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab deploy-image finish
```

Esto concluye el ejercicio guiado.

Administración de aplicaciones con la CLI

Objetivos

Tras finalizar esta sección, usted deberá ser capaz de realizar lo siguiente:

- Implementar una aplicación desde el código fuente y administrar sus recursos mediante la interfaz de línea de comandos.
- Describir los roles de OpenShift requeridos para administrar un proyecto y un clúster.
- Describir cómo Fuente a imagen determina la imagen del compilador para una aplicación.

Privilegios de administración de proyectos y clústeres OpenShift

Muchas de las tareas implicadas en la administración de un clúster OpenShift requieren privilegios administrativos especiales. Puede acceder a un clúster OpenShift si usted es el administrador de clústeres, pero generalmente, como desarrollador, no cuenta con este nivel de acceso.

Los clústeres OpenShift suelen servir a muchos usuarios diferentes, posiblemente de varias organizaciones. Estos clústeres de varios nodos proporcionan a los usuarios diferentes niveles de acceso: administrador de clústeres, administrador de proyecto y desarrollador.

La configuración predeterminada para un clúster OpenShift permite que cualquier usuario cree proyectos nuevos. Un usuario es automáticamente un administrador de proyecto para cualquier proyecto que cree. El administrador del clúster puede modificar los permisos de clúster OpenShift para que los usuarios no tengan permitido crear proyectos. En este escenario, solo un administrador de clústeres puede crear proyectos nuevos. A continuación, el administrador de clústeres asigna privilegios de administrador de proyecto y privilegios de desarrollador a otros usuarios.

En la siguiente lista, se resumen las tareas que pueden realizar los usuarios con cada nivel de acceso:

Administrador de clústeres

Administra proyectos, agrega nodos, crea volúmenes persistentes, asigna cuotas del proyecto y realiza otras tareas de administración en todo el clúster.

Administrador de proyecto

Administra recursos dentro de un proyecto, asigna límites de recursos y otorga a otros usuarios permiso para ver y administrar recursos dentro del proyecto.

Desarrollador

Administra un subconjunto de recursos de un proyecto. El subconjunto incluye recursos requeridos para compilar e implementar aplicaciones, como configuraciones de compilación y de implementación, reclamaciones de volúmenes persistentes, servicios, secretos y rutas. Los desarrolladores no pueden otorgarles a otros usuarios permisos sobre estos recursos ni pueden administrar la mayoría de los recursos al nivel del proyecto, como límites de recursos.

La mayoría de las actividades prácticas de este curso se realizan como usuario desarrollador, que recibe los derechos de administrador de proyecto para los proyectos que crea. Cuando una actividad requiere privilegios de administrador de clústeres, en ella se describe cómo iniciar

sesión como un usuario con privilegios de administrador de clúster y, luego, realizar las tareas administrativas requeridas, o se proporcionan los recursos de administrador preconfigurados.



nota

En el curso *OpenShift Enterprise Administration I* (DO280), se cubren cómo realizar tareas de administración y cómo asignar permisos de administrador de clúster y proyecto a los usuarios.

Solución de problemas en compilaciones, implementaciones y pods mediante la CLI

OpenShift proporciona tres mecanismos primarios para obtener información de solución de problemas acerca de un proyecto y sus recursos:

Información de estado

Comandos como `oc status` y `oc get` brindan información de resumen acerca de los recursos en un proyecto. Use estos comandos para obtener información crítica; por ejemplo, si una compilación falló o si un pod está listo y en ejecución.

Descripción de recursos

El comando `oc describe` muestra información detallada acerca de un recurso, incluido su estado actual, configuración y eventos recientes. La opción `-o` con el comando `oc get` muestra información completa y detallada de configuración y estado acerca de un recurso. Use estos comandos para inspeccionar un recurso y para determinar si OpenShift pudo detectar cualquier condición de error específica relacionada al recurso.

Registros de recursos

Recursos ejecutables, como pods y compilaciones, almacenan registros que se pueden ver con el comando `oc logs`. Estos registros son generados por la aplicación que se ejecuta dentro de un pod o por el proceso de compilación. Use estos comandos para recuperar mensajes de error específicos de la aplicación y para obtener información detallada acerca de errores de compilación.

Cuando estos mecanismos no proporcionan suficiente información, puede usar los comandos `oc cp` y `oc rsh` para interactuar directamente con una aplicación contenedizada.

Comparación de dos comandos que puede usar para describir los recursos de OpenShift

El comando `oc describe` puede seguir relaciones entre recursos. Por ejemplo, describir una configuración de compilación muestra información acerca de las compilaciones más recientes. El comando `oc get -o` muestra información acerca del recurso solicitado únicamente. Por ejemplo, ejecutar el comando `oc get -o` en una configuración de compilación no muestra información acerca de las compilaciones recientes.

Use el comando `oc edit` para realizar cambios en un recurso de OpenShift. El comando `oc edit` combina recuperar la descripción del recurso, con el comando `oc get -o`, abrir el archivo de salida con un editor de texto y, luego, aplicar los cambios con el comando `oc apply`.

Mejora de los registros de aplicaciones contenedorizadas

La usabilidad de los registros de aplicaciones almacenados por OpenShift depende del diseño de la imagen de contenedor de aplicaciones. Se espera que una aplicación contenedizada envíe toda la salida de los registros a la salida estándar. Si la aplicación envía la salida de sus registros a

un archivo de registro, como sucede normalmente con las aplicaciones no contenedorizadas, los registros se conservan dentro del almacenamiento efímero del contenedor y pueden perderse cuando el pod de la aplicación finaliza.

OpenShift también proporciona un subsistema de registro opcional, basado en la pila (stack) *EFK* (Elasticsearch, Fluentd y Kibana). El subsistema de registro brinda almacenamiento a largo plazo y capacidades de búsqueda para los registros de la aplicación y los nodos del clúster OpenShift. Una aplicación puede estar diseñada para sacar provecho del subsistema de registro de OpenShift o para enviar la salida de su registro a la salida estándar y permitir que la pila (stack) EFK recopile y procese sus registros.

La instalación y configuración del subsistema de registro de OpenShift se encuentran fuera del alcance de este curso.

Lectura de registros de compilación

Hay dos maneras para recuperar los registros de compilación para una compilación específica: puede consultar la configuración de compilación o el recurso de compilación, o puede consultar el pod de la compilación.

En el siguiente ejemplo, se usa una configuración de compilación denominada **myapp**:

```
[user@host ~]$ oc logs bc/myapp
```

Los registros de una configuración de compilación son registros de la última compilación, ya sea si se realizó correctamente o no.

En este ejemplo, se usa la segunda compilación de la misma configuración de compilación:

```
[user@host ~]$ oc logs build/myapp-2
```

En este ejemplo, se usa el pod de compilación creado para realizar la misma compilación:

```
[user@host ~]$ oc logs myapp-build-2
```

Obtención de acceso directo a una aplicación contenedorizada

Si una aplicación almacena sus registros en el almacenamiento efímero del contenedor, use los comandos `oc cp` y `oc rsync` para recuperar los archivos de registro. Estos comandos pueden usarse para recuperar cualquier archivo dentro de un sistema de archivos de contenedor en ejecución, como archivos de configuración para la aplicación contenedora.

Debe usar la ruta del archivo remoto en el sistema de archivos de contenedor con los comandos `oc cp` y `oc rsync`. Puede incluir los archivos en el almacenamiento efímero del contenedor o dentro de un volumen persistente montado por el contenedor.

Por ejemplo, para recuperar los registros de errores del servidor HTTP de Apache almacenados dentro del pod de una aplicación denominado **frontend**, use el siguiente comando:

```
[user@host ~]$ oc cp frontend-1-zvjhb:/var/log/httpd/error_log \
> /tmp/frontend-server.log
```

El comando `oc cp` copia todas las carpetas de manera predeterminada. Si el argumento de origen es un único archivo, el argumento de destino también necesita ser un único archivo. A diferencia

del comando UNIX cp, el comando oc cp no puede copiar un archivo de origen a una carpeta de destino.

El comando oc cp requiere que la imagen del contenedor de aplicaciones subyacente proporcione el comando tar para funcionar. Si tar no está instalado dentro del contenedor de aplicaciones, oc cp fallará.

El mismo comando se usa para copiar archivos a un sistema de archivos del contenedor. Use esta capacidad para realizar pruebas rápidas dentro de un contenedor en ejecución. No use esta capacidad para corregir un problema de forma permanente. La manera recomendada para corregir un problema en un contenedor es aplicar la corrección a la imagen de contenedor y a los recursos de la aplicación y, luego, implementar un nuevo pod de aplicación.

El comando oc rsync sincroniza las carpetas locales con carpetas remotas de un contenedor en ejecución. Usa el comando rsync local para reducir el uso del ancho de banda, pero no requiere que los comandos rsync o ssh estén disponibles en la imagen de contenedor.

Si no es suficiente con recuperar los archivos para solucionar el problema de un contenedor en ejecución, el comando oc rsh crea una shell remota para ejecutar comandos dentro del contenedor. Usa la API maestra de OpenShift para crear un túnel seguro al pod remoto, pero no usa los comandos ssh o rsh de UNIX.

En el siguiente ejemplo, se demuestra la ejecución del comando ps ax dentro de un pod denominado frontend:

```
[user@host ~]$ oc rsh frontend-1-zvjhb ps ax
```



nota

Muchas imágenes de contenedor no incluyen comandos de solución de problemas de UNIX, como ps y ping. El comando oc rsh solo puede ejecutar comandos que estén proporcionados por el contenedor remoto.

Agregue la opción -t al comando oc rsh para iniciar una sesión de shell interactiva dentro del contenedor:

```
[user@host ~]$ oc rsh -t frontend-1-zvjhb
```



nota

El prompt de shell que se muestra con el comando oc rsh depende de la shell proporcionada por la imagen de contenedor.

Variables del entorno de compilación e implementación

Muchas imágenes de contenedor esperan que los usuarios definan las variables del entorno para proporcionar datos de configuración. Por ejemplo, la imagen de la base de datos MySQL de Red Hat Container Catalog requiere la variable MYSQL_DATABASE para proporcionar el nombre de la base de datos.

Agregue la opción `-e` al comando `oc new-app` para proporcionar valores para las variables del entorno. Estos valores están almacenados en la configuración de implementación y se agregan a todos los pods creados por una implementación.

Las imágenes de compilador Source-to-Image (Fuente a imagen, S2I) también pueden aceptar parámetros de configuración de las variables del entorno. Por ejemplo, las aplicaciones Node.js a menudo requieren un repositorio `npm`, el administrador de paquetes principal para Node.js, para descargar las dependencias de Node.js requeridas por la aplicación. Por este motivo, el compilador Node.js S2I de Container Catalog acepta la variable `npm_config_registry` o `NPM_PROXY` para proporcionar una URL en la que el compilador S2I puede ubicar el repositorio de módulos npm requerido para recuperar las dependencias de Node.js necesarias.



nota

El comando `npm`, ejecutado por la imagen del compilador Node.js S2I, requiere que proporcione un valor para la variable de entorno `npm_config_registry`. El script `assemble` de la imagen del compilador Node.js S2I, que llama al comando `npm`, requiere que proporcione un valor para la variable de entorno `NPM_PROXY`.

Las variables de la imagen del compilador S2I son útiles para evitar el almacenamiento de información de configuración con las fuentes de la aplicación, en el repositorio Git. Diferentes entornos podrían requerir diferentes opciones de configuración; por ejemplo:

- Un entorno de desarrollo usaría un servidor de repositorio npm donde los desarrolladores puedan instalar módulos nuevos.
- Un entorno de control de calidad usará un servidor de repositorio npm diferente, donde un equipo de seguridad podría analizar los módulos antes de promoverlos a entornos superiores.

Usted define las variables de entorno para un pod de aplicación mediante la opción `-e` del comando `oc new-app`. Para un pod de compilador, se definen las variables de entorno mediante la opción `--build-env` del comando `oc new-app`.

Tenga en cuenta que una configuración de implementación almacena variables de entorno para pods de la aplicación, mientras que una configuración de compilación almacena las variables de entorno para pods del compilador. Consulte los documentos de cada imagen del compilador para obtener información acerca de sus variables de compilación y sus valores predeterminados.



Referencias

Para obtener más información, consulte el capítulo *Análisis de contenedores, imágenes y flujos de imágenes* de la guía *Imágenes para Red Hat OpenShift Container Platform 4.5* en
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/images/understanding-images

► Ejercicio Guiado

Administración de una aplicación con la CLI

En este ejercicio, implementará una aplicación contenedora compuesta de varios pods desde una plantilla. También usará la solución de problemas para corregir un error de implementación.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear una aplicación desde una plantilla personalizada. La plantilla implementa un pod de aplicación a partir de código fuente PHP y un pod de base de datos a partir de una imagen de contenedor del servidor MySQL.
- Identificar la causa raíz de un error de implementación con OpenShift CLI.
- Corregir el error con OpenShift CLI.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I y la imagen de la base de datos requeridas por la plantilla personalizada (PHP 7.2 y MySQL 5.7).
- La aplicación de muestra en el repositorio Git (quotes).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab build-template start
```

► 1. Revise el código fuente de la aplicación Quotes.

- 1.1. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación `master` del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd ~/D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 1.2. La aplicación está compuesta por dos páginas PHP:

```
[student@workstation D0288-apps]$ ls ~/D0288-apps/quotes
get.php  index.php
```

En la página de bienvenida (`index.php`), se muestra una breve descripción de la aplicación. Se vincula a otra página (`get.php`) que obtiene una cotización aleatoria de una base de datos MySQL.

- 1.3. Revise el código PHP que accede a la base de datos:

```
[student@workstation D0288-apps]$ less ~/D0288-apps/quotes/get.php
<?php
    $link = mysqli_connect($_ENV["DATABASE_SERVICE_NAME"], $_ENV["DATABASE_USER"],
    $_ENV["DATABASE_PASSWORD"], $_ENV["DATABASE_NAME"]);
    if (!$link) {
        http_response_code(500);
        error_log("Error: unable to connect to database\n");
        die();
    }
...output omitted...
```

Presione q para salir.

La misma aplicación solo usa funciones PHP estándares y no usa marco (framework). Usa variables de entorno para recuperar los parámetros de conexión de la base de datos y devuelve códigos de estado HTTP estándares si hay errores.

- 2. Inspeccione la plantilla personalizada en `~/D0288/labs/build-template/php-mysql-ephemeral.json`. No se enumera el contenido completo de la plantilla personalizada para ahorrar espacio. No necesita realizar modificaciones en la plantilla; está lista para su utilización. En los siguientes listados, se revisan las partes más importantes de la plantilla:

- 2.1. La plantilla comienza definiendo un secreto y una ruta:

```
{
    "kind": "Template",
    "apiVersion": "v1",
    "metadata": {
        "name": "php-mysql-ephemeral", ①
    ...output omitted...
    "objects": [
        {
            "apiVersion": "v1",
            "kind": "Secret", ②
    ...output omitted...
        },
        "stringData": {
            "database-password": "${DATABASE_PASSWORD}",
            "database-user": "${DATABASE_USER}"
    ...output omitted...
        },
        "apiVersion": "v1",
        "kind": "Route", ③
    ...output omitted...
    "spec": {
        "host": "${APPLICATION_DOMAIN}",
        "to": {
```

```

        "kind": "Service",
        "name": "${NAME}"
...output omitted...

```

- ➊ La plantilla es una copia de la plantilla cakephp-mysql-example estándar, con recursos y parámetros específicos de ese marco (framework) eliminado. La plantilla personalizada es adecuada para cualquier aplicación PHP simple que use una base de datos MySQL.
 - ➋ Un secreto almacena credenciales de inicio de sesión de la base de datos y completa las variables de entorno en los pods de la aplicación y la base de datos. Explicaremos los secretos de OpenShift más adelante en este manual.
 - ➌ Una ruta proporciona acceso externo a la aplicación.
- 2.2. La plantilla define los recursos para una aplicación PHP. En el siguiente listado, se hace hincapié en la configuración de compilación y se omiten los recursos de flujo de imágenes y servicios:

```

...output omitted...
{
    "apiVersion": "v1",
    "kind": "BuildConfig", ➊
...output omitted...
    "source": {
        "contextDir": "${CONTEXT_DIR}",
        "git": {
            "ref": "${SOURCE_REPOSITORY_REF}",
            "uri": "${SOURCE_REPOSITORY_URL}"
        },
        "type": "Git"
    },
    "strategy": {
        "sourceStrategy": {
            "from": {
                "kind": "ImageStreamTag", ➋
                "name": "php:7.2",
            }
        }
    }
...output omitted...

```

- ➊ Una configuración de compilación usa el proceso S2I para compilar e implementar una aplicación PHP del código fuente. La configuración de compilación y los recursos asociados son los mismos que se crearían con el comando `oc new-app` en el repositorio Git.
 - ➋ Un flujo de imágenes OpenShift estándar proporciona la imagen del compilador de tiempo de ejecución PHP.
- 2.3. La plantilla define los recursos para una base de datos MySQL. El siguiente listado se centra en la configuración de implementación y omite los recursos de flujo de imágenes y servicios:

```

...output omitted...
{
    "apiVersion": "v1",
    "kind": "DeploymentConfig", ➊
...output omitted...
    "containers": [

```

```

...output omitted...
        "name": "mysql",
        "ports": [
            {
                "containerPort": 3306
...output omitted...
        "volumes": [
            {
                "emptyDir": {}, ❷
                "name": "data"
...output omitted...
        "triggers": [
...output omitted...
        "from": {
            "kind": "ImageStreamTag", ❸
            "name": "mysql:5.7",
...output omitted...

```

- ❶ Una configuración de implementación implementa un contenedor de la base de datos MySQL. La configuración de implementación y los recursos asociados son los mismos que se crearán con el comando `oc new-app --as-deployment-config` en la imagen de base de datos.
- ❷ La base de datos no está respaldada por almacenamiento persistente. Se podrían perder todos los datos si se reinicia el pod de la base de datos.
- ❸ Un flujo de imágenes OpenShift estándar proporciona la imagen de base de datos MySQL.

2.4. Por último, la plantilla define algunos parámetros. A continuación, se detallan algunos de estos parámetros. Usará más de ellos.

```

...output omitted...
    "parameters": [
        {
            "name": "NAME",
            "displayName": "Name",
            "description": "The name assigned to all of the app objects defined in
this template.",
...output omitted...
        {
            "name": "SOURCE_REPOSITORY_URL", ❶
            "displayName": "Git Repository URL",
            "description": "The URL of the repository with your application source
code.",
...output omitted...
        {
            "name": "DATABASE_USER", ❷
            "displayName": "Database User",
...output omitted...

```

- ❶ Los parámetros proporcionan configuraciones específicas de la aplicación, como la URL del repositorio Git.
- ❷ Los parámetros de la plantilla también incluyen la configuración de la base de datos y las credenciales de conexión.

► **3.** Instale la plantilla personalizada.

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 D0288-apps]$ source /usr/local/etc/ocp4.config
```

3.2. Inicie sesión en OpenShift con su nombre de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

3.3. Busque una plantilla que use PHP y MySQL. OpenShift proporciona una plantilla basada en el marco (framework) CakePHP.

Esta plantilla no es adecuada para la aplicación Quotes porque agrega recursos y dependencias requeridos por el marco (framework). Se proporciona una plantilla más simple para este ejercicio; la plantilla que revisó en el paso anterior.

```
[student@workstation D0288-apps]$ oc get templates -n openshift | grep php \
> | grep mysql
cakephp-mysql-example      An example CakePHP application ...
cakephp-mysql-persistent    An example CakePHP application ...
```

3.4. Cree un nuevo proyecto para alojar la plantilla personalizada:

```
[student@workstation D0288-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-common
Now using project "youruser-common" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
[student@workstation D0288-apps]$ oc create -f \
> ~/D0288/labs/build-template/php-mysql-ephemeral.json
template.template.openshift.io/php-mysql-ephemeral created
```

Red Hat le recomienda crear recursos OpenShift reutilizables, como flujos de imágenes y plantillas, en un proyecto compartido.

► **4.** Implemente la aplicación mediante la plantilla personalizada.

4.1. Cree un nuevo proyecto para alojar la aplicación:

```
[student@workstation D0288-apps]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-build-template
Now using project "youruser-build-template" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

4.2. Revise los parámetros de la plantilla para decidir cuáles se podrían requerir para implementar la aplicación Quotes. Lea la descripción de los parámetros de la plantilla:

```
[student@workstation D0288-apps]$ oc describe template php-mysql-ephemeral \
> -n ${RHT_OCP4_DEV_USER}-common
Name: php-mysql-ephemeral
...output omitted...
Parameters:
  Name:           NAME
  Display Name:  Name
  Description:   The name assigned to all of the app objects defined in this
template.
  Required:      true
  Value:         php-app
...output omitted...
```

- 4.3. Revise el script `create-app.sh`. Proporciona el comando `oc new-app`, que usa la plantilla personalizada y brinda todos los parámetros requeridos para implementar la aplicación Quotes, de modo que no tiene que escribir un comando muy largo:

```
[student@workstation D0288-apps]$ cat ~/D0288/labs/build-template/create-app.sh
...output omitted...
oc new-app --as-deployment-config --template ${RHT_OCP4_DEV_USER}-common/php-
mysql-ephemeral \
-p NAME=quotesapi \
-p APPLICATION_DOMAIN=quote-${RHT_OCP4_DEV_USER}.${RHT_OCP4_WILDCARD_DOMAIN} \
-p SOURCE_REPOSITORY_URL=https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
-p CONTEXT_DIR=quotes \
-p DATABASE_SERVICE_NAME=quotesdb \
-p DATABASE_USER=user1 \
-p DATABASE_PASSWORD=mypa55 \
--name quotes
```

- 4.4. Ejecute el script `create-app.sh`:

```
[student@workstation D0288-apps]$ ~/D0288/labs/build-template/create-app.sh
--> Deploying template "youruser-common/php-mysql-ephemeral" for "youruser-common/
php-mysql-ephemeral" to project youruser-build-template
...output omitted...
--> Creating resources ...
  secret "quotesapi" created
  service "quotesapi" created
  route.route.openshift.io "quotesapi" created
  imagestream.image.openshift.io "quotesapi" created
  buildconfig.build.openshift.io "quotesapi" created
  deploymentconfig.apps.openshift.io "quotesapi" created
  service "quotesdb" created
  deploymentconfig.apps.openshift.io "quotesdb" created
--> Success
...output omitted...
```

- 4.5. Siga los registros de compilación de la aplicación:

```
[student@workstation D0288-apps]$ oc logs -f bc/quotesapi
Cloning "https://github.com/youruser/D0288-apps" ...
...output omitted...
Push successful
```

- 4.6. Espere a que el pod de la aplicación y de la base de datos esté listo y en ejecución. Tenga en cuenta que los nombres exactos de sus pods pueden diferir de los que se muestran en la siguiente salida:

```
[student@workstation D0288-apps]$ oc get pod
NAME          READY   STATUS    RESTARTS   AGE
quotesapi-1-build  0/1     Completed   0          89s
quotesapi-1-deploy  0/1     Completed   0          35s
quotesapi-1-r6f31   1/1     Running    0          28s
quotesdb-1-deploy  0/1     Completed   0          88s
quotesdb-1-hh2g9    1/1     Running    0          80s
```

Tome nota de los nombres del pod de la aplicación y la base de datos (quotesapi-1-r6f31 y quotesdb-1-hh2g9 en la salida de muestra). Los necesitará para los próximos pasos.

- 4.7. Use la ruta creada por la plantilla para probar el extremo (endpoint) /get.php de la aplicación. Devuelve un código de error HTTP:

```
[student@workstation D0288-apps]$ oc get route
NAME      HOST/PORT
quotesapi  quote-youruser.apps.cluster.domain.example.com  ...
[student@workstation D0288-apps]$ curl -si \
> http://quote-$RHT_OCP4_DEV_USER.$RHT_OCP4_WILDCARD_DOMAIN/get.php
HTTP/1.1 500 Internal Server Error
...output omitted...
```

- 5. Aplique la solución de problemas en la conectividad entre el pod de la aplicación y la base de datos.

La aplicación no funciona de la manera esperada, pero los procesos de compilación e implementación se realizaron correctamente. El error de la aplicación puede deberse a un defecto de la aplicación, un requisito previo ausente o una configuración incorrecta.

Como primer paso en la solución de problemas, verifique que el pod de la aplicación esté conectado al pod de la base de datos correcto.

- 5.1. Verifique que el servicio de la base de datos haya encontrado el pod de la base de datos correcto. Use el nombre del pod de la base de datos que obtuvo de *Paso 4.6*:

```
[student@workstation D0288-apps]$ oc describe svc quotesdb | grep Endpoints
Endpoints:  10.129.0.142:3306
[student@workstation ~]$ oc describe pod quotesdb-1-hh2g9 | grep IP
IP:  10.129.0.142
```

- 5.2. Verifique las credenciales de inicio de sesión del pod de la base de datos:

```
[student@workstation D0288-apps]$ oc describe pod quotesdb-1-hh2g9 \
> | grep -A 4 Environment
Environment:
  MYSQL_USER:      <set to the key 'database-user' in secret 'quotesapi'>
  MYSQL_PASSWORD:  <set to the key 'database-password' in secret 'quotesapi'>
  MYSQL_DATABASE:  phpapp
Mounts:
```

- 5.3. Verifique los parámetros de conexión de la base de datos en el pod de la aplicación.
Use el nombre del pod de la aplicación que obtuvo de *Paso 4.6*:

```
[student@workstation D0288-apps]$ oc describe pod quotesapi-1-r6f31 \
> | grep -A 5 Environment
Environment:
  DATABASE_SERVICE_NAME:   quotesdb
  DATABASE_NAME:          phpapp
  DATABASE_USER:          <set to the key 'database-user' in secret
                           'quotesapi'>
  DATABASE_PASSWORD:       <set to the key 'database-password' in secret
                           'quotesapi'>
Mounts:
```

Observe que las variables de entorno que proporcionan las credenciales de inicio de sesión de la base de datos coinciden entre los dos pods:

- **DATABASE_NAME** es igual al valor de **MYSQL_DATABASE**.
- **DATABASE_USER** se establece en el valor de la clave **database-user** en el secreto **quotesapi**.
- **DATABASE_PASSWORD** se establece en el valor de la clave **database-user** en el secreto **quotesapi**.
- **DATABASE_SERVICE_NAME** es igual al nombre de servicio de la base de datos, que es **quotesdb**.

- 5.4. Verifique que el pod de la aplicación pueda comunicarse con el pod de la base de datos. La imagen del compilador S2I de PHP no proporciona utilidades de red comunes, como el comando **ping**, pero en este caso el comando **curl** proporciona una salida útil:

```
[student@workstation D0288-apps]$ oc rsh quotesapi-1-r6f31 bash -c \
> 'echo > /dev/tcp/$DATABASE_SERVICE_NAME/3306 && echo OK || echo FAIL'
OK
```

La salida del comando anterior demuestra que la conectividad de red no es el problema.

- 6. Revise los registros de aplicación para encontrar la causa raíz del error y corregirlo.

- 6.1. Revise los registros de la aplicación.

```
[student@workstation D0288-apps]$ oc logs quotesapi-1-r6f31
AH00558: httpd: Could not reliably determine the server's fully qualified domain
name, using 10.129.0.143. Set the 'ServerName' directive globally to suppress
this message
...output omitted...
[Mon May 27 14:54:56.187516 2019] [php7:notice] [pid 52] [client
10.128.2.3:43952] SQL error: Table 'phpapp.quote' doesn't exist\n
10.128.2.3 - - [27/May/2019:14:54:51 +0000] "GET /get.php HTTP/1.1" 500 - "-"
"curl/7.29.0"
...output omitted...
```

El mensaje acerca del nombre del servidor se puede ignorar sin consecuencias. En la entrada del registro que precede el código del error HTTP, se muestra un error SQL. El error SQL indica que la aplicación no puede consultar la tabla quote en la base de datos phpapp.

El paso anterior indicó que el nombre de la base de datos es correcto. La conclusión lógica es que no se creó la tabla. Se proporciona un script SQL para completar la base de datos como parte de los archivos de este ejercicio, en la carpeta ~/D0288/labs/build-template.

- 6.2. Copie el script SQL en el pod de la base de datos:

```
[student@workstation D0288-apps]$ oc cp ~/D0288/labs/build-template/quote.sql \
> quotesdb-1-hh2g9:/tmp/quote.sql
```

- 6.3. Ejecute el script SQL dentro del pod de la base de datos:

```
[student@workstation D0288-apps]$ oc rsh -t quotesdb-1-hh2g9
sh-4.2$ mysql -u$MYSQL_USER -p$MYSQL_PASSWORD $MYSQL_DATABASE < /tmp/quote.sql
...output omitted...
sh-4.2$ exit
[student@workstation D0288-apps]$
```

- 6.4. Acceda a la aplicación para verificar que ahora funciona. Recibirá un presupuesto aleatorio. Recuerde usar el nombre de host de la ruta de *Paso 4.6*:

```
[student@workstation D0288-apps]$ curl -si \
> http://quote-$RHT_OCP4_DEV_USER.$RHT_OCP4_WILDCARD_DOMAIN/get.php
HTTP/1.1 200 OK
...output omitted...
Always remember that you are absolutely unique. Just like everyone else.
```

Probablemente verá un mensaje aleatorio diferente, pero si recibe una cotización, podrá comprobar que la aplicación ahora funciona.

- ▶ 7. Realice la limpieza. Elimine los proyectos creados durante este ejercicio de la carpeta principal.

```
[student@workstation DO288-apps]$ cd ~  
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-build-template  
project.project.openshift.io "youruser-build-template" deleted  
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-common  
project.project.openshift.io "youruser-common" deleted
```

Finalizar

En `workstation`, ejecute el comando `lab build-template finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab build-template finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Implementación y administración de aplicaciones en un clúster OpenShift

Listado de verificación de rendimiento

En este trabajo de laboratorio, implementará una aplicación en un clúster OpenShift a partir de código fuente. El archivo de configuración de la compilación de aplicaciones tiene un error que deberá encontrar y corregir.



nota

El comando `grade` al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se indica en la especificación del trabajo de laboratorio.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear una aplicación mediante la estrategia de compilación source.
- Consultar los registros de compilación para encontrar información acerca del error de compilación.
- Actualizar la configuración de la herramienta de compilación de la aplicación para corregir el error de compilación.
- Volver a compilar la aplicación y verificar que se implemente correctamente.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I para las aplicaciones Node.js 12.
- La aplicación en el repositorio Git (`nodejs-helloworld`).

Ejecute el siguiente comando en `workstation` para validar los requisitos previos. El comando también descarga archivos auxiliares y de solución para el trabajo de laboratorio de revisión:

```
[student@workstation ~]$ lab source-build start
```

Requisitos

La aplicación proporcionada se escribe en JavaScript con el tiempo de ejecución Node.js. Es una aplicación "hello, world" basada en el marco (framework) Express. Compile e implemente la aplicación en un clúster OpenShift, según los siguientes requisitos:

- El código de aplicación se implementa desde una nueva bifurcación, denominada `source-build`.
 - El nombre del proyecto para OpenShift es `youruser-source-build`.
 - El nombre de la aplicación para OpenShift es `greet`.
 - Se debe poder acceder a la aplicación desde la ruta predeterminada:
`greet-youruser-source-build.apps.cluster.domain.example.com`
 - Este es el repositorio Git que contiene las fuentes del directorio de aplicaciones:
`https://github.com/yourgithubuser/D0288-apps/nodejs-helloworld`.
 - Los módulos npm requeridos para compilar la aplicación están disponibles en:
`http://nexus-common.apps.cluster.domain.example.com/repository/nodejs`
- Use la variable de entorno de compilación `npm_config_registry` para enviar esta información a la imagen del compilador S2I para Node.js.
- Puede usar el comando `python -m json.tool filename.json` para identificar los errores de sintaxis en archivos JSON.

Pasos

1. Navegue a su clon local del repositorio Git D0288-apps y cree una nueva bifurcación denominada `source-build` de la bifurcación maestra. Implemente la aplicación en la carpeta `nodejs-helloworld` en el proyecto `youruser-source-build` del clúster OpenShift.
2. Visualice los registros de compilación para identificar el error de compilación e inspeccione las fuentes de la aplicación para determinar la causa raíz.
Recuerde que puede usar el comando `python -m json.tool filename` para validar un archivo JSON.
3. Corrija el error en el archivo de configuración de herramientas de compilación y envíe los cambios al repositorio Git.
4. Inicie una nueva compilación de la aplicación y verifique que la aplicación se implemente correctamente.
5. Verifique que los registros de aplicaciones no muestren errores, exponga la aplicación para acceso externo y verifique que la aplicación devuelva un mensaje "hello, world".

Evaluación

Inicie sesión con el usuario `student` en la máquina `workstation` y use el comando `lab` para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

```
[student@workstation ~]$ lab source-build grade
```

Finalizar

En `workstation`, ejecute el comando `lab source-build finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab source-build finish
```

Esto concluye el trabajo de laboratorio.

► Solución

Implementación y administración de aplicaciones en un clúster OpenShift

Listado de verificación de rendimiento

En este trabajo de laboratorio, implementará una aplicación en un clúster OpenShift a partir de código fuente. El archivo de configuración de la compilación de aplicaciones tiene un error que deberá encontrar y corregir.



nota

El comando `grade` al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se indica en la especificación del trabajo de laboratorio.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear una aplicación mediante la estrategia de compilación source.
- Consultar los registros de compilación para encontrar información acerca del error de compilación.
- Actualizar la configuración de la herramienta de compilación de la aplicación para corregir el error de compilación.
- Volver a compilar la aplicación y verificar que se implemente correctamente.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I para las aplicaciones Node.js 12.
- La aplicación en el repositorio Git (`nodejs-helloworld`).

Ejecute el siguiente comando en `workstation` para validar los requisitos previos. El comando también descarga archivos auxiliares y de solución para el trabajo de laboratorio de revisión:

```
[student@workstation ~]$ lab source-build start
```

Requisitos

La aplicación proporcionada se escribe en JavaScript con el tiempo de ejecución Node.js. Es una aplicación “hello, world” basada en el marco (framework) Express. Compile e implemente la aplicación en un clúster OpenShift, según los siguientes requisitos:

- El código de aplicación se implementa desde una nueva bifurcación, denominada `source-build`.
 - El nombre del proyecto para OpenShift es `youruser-source-build`.
 - El nombre de la aplicación para OpenShift es `greet`.
 - Se debe poder acceder a la aplicación desde la ruta predeterminada:
`greet-youruser-source-build.apps.cluster.domain.example.com`
 - Este es el repositorio Git que contiene las fuentes del directorio de aplicaciones:
`https://github.com/yourgithubuser/D0288-apps/nodejs-helloworld`.
 - Los módulos npm requeridos para compilar la aplicación están disponibles en:
`http://nexus-common.apps.cluster.domain.example.com/repository/nodejs`
- Use la variable de entorno de compilación `npm_config_registry` para enviar esta información a la imagen del compilador S2I para Node.js.
- Puede usar el comando `python -m json.tool filename.json` para identificar los errores de sintaxis en archivos JSON.

Pasos

1. Navegue a su clon local del repositorio Git D0288-apps y cree una nueva bifurcación denominada `source-build` de la bifurcación maestra. Implemente la aplicación en la carpeta `nodejs-helloworld` en el proyecto `youruser-source-build` del clúster OpenShift.

- 1.1. Prepare el entorno del trabajo de laboratorio.

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

- 1.2. Inicie sesión en OpenShift y cree el proyecto:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-source-build
...output omitted...
```

- 1.3. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación `master` (maestra) del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 1.4. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b source-build
Switched to a new branch 'source-build'
[student@workstation D0288-apps]$ git push -u origin source-build
...output omitted...
* [new branch]      source-build -> source-build
Branch source-build set up to track remote branch source-build from origin.
```

- 1.5. Cree una nueva aplicación desde las fuentes del repositorio Git. Asígnele el nombre `greet` a la aplicación. Use la opción `--build-env` con el comando `oc new-app --as-deployment-config` para definir la variable del entorno de compilación con la URL de los módulos npm.

Copie o ejecute el comando del script `oc-new-app.sh` en la carpeta `/home/student/D0288/labs/source-build`:

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config --name greet \
> --build-env npm_config_registry=\
> http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs \
> nodejs:12-https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#source-build \
> --context-dir nodejs-helloworld
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "greet" created
buildconfig.build.openshift.io "greet" created
deploymentconfig.apps.openshift.io "greet" created
service "greet" created
--> Success
...output omitted...
```



nota

No hay espacio antes ni después del signo igual (=) después de `npm_config_registry`. El par `key=value` completo para la variable de entorno de compilación es demasiado largo para el ancho de la página.

2. Visualice los registros de compilación para identificar el error de compilación e inspeccione las fuentes de la aplicación para determinar la causa raíz.

Recuerde que puede usar el comando `python -m json.tool filename` para validar un archivo JSON.

- 2.1. Siga los registros de compilación:

```
[student@workstation D0288-apps]$ oc logs -f bc/greet
```

Debe ver un mensaje de error de análisis JSON:

```
...output omitted...
--> Installing all dependencies
npm ERR! code EJSONPARSE
npm ERR! file /opt/app-root/src/package.json
npm ERR! JSON.parse Failed to parse json
npm ERR! JSON.parse Unexpected string in JSON at position 241 while parsing '{
```

```
npm ERR! JSON.parse  "name": "nodejs-helloworld",
npm ERR! JSON.parse  "vers"
npm ERR! JSON.parse Failed to parse package.json data.
npm ERR! JSON.parse package.json must be actual JSON, not just JavaScript.
...output omitted...
```

La imagen del compilador Node.js no proporciona una ubicación específica del error dentro del archivo de configuración de las herramientas de compilación package.json.

- 2.2. Use el comando `python -m json.tool` para validar el archivo JSON:

```
[student@workstation D0288-apps]$ python -m json.tool \
> nodejs-helloworld/package.json
```

Debe ver el siguiente mensaje de error:

```
Expecting : delimiter: line 12 column 15 (char 241)
```

- 2.3. Abra el archivo de configuración de herramientas de compilación ~/D0288-apps/nodejs-helloworld/package.json con un editor de texto e identifique el error de sintaxis. En la siguiente lista parcial, se muestra que faltan dos puntos (:) después de la clave express:

```
"dependencies": {
  "express" "4.14.x"
}
```

3. Corrija el error en el archivo de configuración de herramientas de compilación y envíe los cambios al repositorio Git.
 - 3.1. Edite el archivo fuente ~/D0288-apps/nodejs-helloworld/package.json para agregar dos puntos (:) después de la clave express. El contenido final del archivo debe ser el siguiente:

```
"dependencies": {
  "express": "4.14.x"
}
```

- 3.2. Confirme y envíe las correcciones al repositorio Git:

```
[student@workstation D0288-apps]$ cd nodejs-helloworld
[student@workstation nodejs-helloworld]$ git commit -a -m 'Fixed JSON syntax'
...output omitted...
[student@workstation nodejs-helloworld]$ git push
...output omitted...
[student@workstation nodejs-helloworld]$ cd ~
[student@workstation ~]$
```

4. Inicie una nueva compilación de la aplicación y verifique que la aplicación se implemente correctamente.
 - 4.1. Inicie una nueva compilación de la aplicación greet y siga sus registros. Espere a que la compilación finalice sin errores:

```
[student@workstation ~]$ oc start-build --follow bc/greet
build "greet-2" started
...output omitted...
Push successful
```

- 4.2. Verifique que se inicie una nueva implementación:

```
[student@workstation ~]$ oc status
...output omitted...
svc/greet - 172.30.160.185:8080
dc/greet deploys istag/greet:latest <-
bc/greet source builds https://github.com/yourgithubuser/D0288-apps#source-
build on openshift/nodejs:10
  deployment #1 deployed 23 seconds ago - 1 pod
...output omitted...
```

- 4.3. Espere hasta que el pod de aplicación esté listo y en ejecución:

| NAME | READY | STATUS | RESTARTS | AGE |
|----------------------|------------|----------------|----------|------------|
| greet-1-build | 0/1 | Error | 0 | 4m |
| greet-1-deploy | 0/1 | Completed | 0 | 14m |
| greet-2-build | 0/1 | Completed | 0 | 2m |
| greet-1-gf59d | 1/1 | Running | 0 | 59s |

5. Verifique que los registros de aplicaciones no muestren errores, exponga la aplicación para acceso externo y verifique que la aplicación devuelva un mensaje “hello, world”.

- 5.1. Acceda a los registros de la aplicación. No debe ver mensajes de error de la aplicación.

```
[student@workstation ~]$ oc logs greet-1-gf59d
...output omitted...
Example app listening on port 8080!
```

- 5.2. Exponga la aplicación a acceso externo:

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

- 5.3. Obtenga el nombre de host generado por OpenShift para la nueva ruta:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT          ...
greet     greet-youruser-source-build.apps.cluster.domain.example.com ...
```

- 5.4. Envíe una solicitud HTTP a la aplicación mediante el comando curl y el nombre de host del paso anterior. Debe devolver un mensaje “hello, world”:

```
[student@workstation ~]$ curl \
> http://greet-${RHT_OCP4_DEV_USER}-source-build.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World!
```

Evaluación

Inicie sesión con el usuario `student` en la máquina `workstation` y use el comando `lab` para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

```
[student@workstation ~]$ lab source-build grade
```

Finalizar

En `workstation`, ejecute el comando `lab source-build finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab source-build finish
```

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- RHOCP proporciona una herramienta de PaaS que se ejecuta en Red Hat CoreOS y Kubernetes.
- OpenShift puede compilar imágenes de contenedor a partir del código fuente de la aplicación o directamente desde Dockerfiles mediante el proceso S2I.
- Los recursos de configuración de compilación e implementación automatizan los procesos de compilación e implementación, y pueden reaccionar automáticamente a los cambios en el código fuente de la aplicación o las actualizaciones realizadas en las imágenes de contenedor.
- El comando `oc new-app` puede detectar automáticamente el lenguaje de programación de origen usado por una aplicación en un repositorio Git. También proporciona varias opciones de desambiguación.
- Los subcomandos `oc` útiles para la solución de problemas en compilaciones e implementaciones son `get`, `describe`, `edit`, `logs`, `cp` y `rsh`.
- Es posible que los desarrolladores no tengan privilegios de administrador de clúster para su entorno de desarrollo, o bien que solo tengan privilegios de administración de proyectos o de edición en un subconjunto de todos los proyectos en el clúster OpenShift.

Diseño de aplicaciones contenerizadas para OpenShift

Meta

Seleccionar un método de organización de aplicaciones contenedorizadas para una aplicación y empaquetarla para que se ejecute en un clúster OpenShift.

Objetivos

- Seleccionar un método adecuado de organización de aplicaciones contenedorizadas.
- Compilar una imagen de contenedor con directivas avanzadas de Dockerfile.
- Seleccionar un método para insertar datos de configuración en una aplicación y crear los recursos necesarios para hacerlo.

Secciones

- Selección de un enfoque de contenedорización (y cuestionario)
- Compilación de imágenes de contenedor con directivas avanzadas de Dockerfile (y ejercicio guiado)
- Inserción de datos de configuración en una aplicación (y ejercicio guiado)

Trabajo de laboratorio

Diseño de aplicaciones contenerizadas para OpenShift

Selección de un enfoque de contenerización

Objetivos

Tras finalizar esta sección, deberá ser capaz de seleccionar un método adecuado para organizar aplicaciones contenedorizadas.

Selección de un método de compilación

La unidad de implementación primaria de OpenShift es una *imagen de contenedor*, también conocida como *imagen a secas*. Una imagen de contenedor consta de la aplicación y todas sus dependencias, como librerías compartidas, entornos de tiempo de ejecución, intérpretes, etc. Hay varias maneras de crear imágenes de contenedor, dependiendo del tipo de aplicación que planea implementar y ejecutar en un clúster OpenShift:

Imágenes de contenedores

Las imágenes de contenedor compiladas fuera de OpenShift se pueden implementar directamente en un clúster OpenShift. Este enfoque es útil cuando ya tiene una aplicación en un paquete como imagen de contenedor. También se usa cuando tiene una imagen de contenedor certificada y soportada, proporcionada por un proveedor externo. Puede implementar imágenes compiladas por un proveedor externo en un clúster OpenShift.

Dockerfiles

En algunos casos, se le proporciona el Dockerfile usado para compilar la imagen de contenedor de aplicaciones. En dichos escenarios, tiene más opciones para tener en cuenta:

- Puede personalizar el Dockerfile y compilar nuevas imágenes para satisfacer las necesidades de su aplicación. Use esta opción cuando los cambios son pequeños y si no desea agregar demasiadas capas a la imagen.
- Puede crear un Dockerfile con la imagen de contenedor provista como principal y personalizar la imagen de base para adecuarse a las necesidades de su aplicación. Use esta opción si desea crear una nueva imagen secundaria con más personalizaciones y que herede las capas de la imagen principal.

Imágenes de compilador de fuente a imagen (S2I)

Una imagen de compilador S2I contiene librerías del sistema operativo base, compiladores e intérpretes, tiempos de ejecución, marcos (frameworks) y herramientas de fuente a imagen. Cuando se crea una aplicación mediante este enfoque, OpenShift combina el código fuente de la aplicación y la imagen de compilador para crear una imagen de contenedor que OpenShift pueda implementar en el clúster. Este enfoque tiene varias ventajas para los desarrolladores y es la manera más rápida de compilar nuevas aplicaciones para implementaciones en un clúster OpenShift.

Según las necesidades de su aplicación, tiene diferentes opciones para usar imágenes de compilador S2I:

- Red Hat proporciona muchas imágenes de compilador S2I soportadas para diferentes tipos de aplicaciones. Red Hat le recomienda usar una de estas imágenes de compilador S2I estándar cuando sea posible.
- Las imágenes de compilador S2I son imágenes de contenedor simples, pero con metadatos, scripts y herramientas adicionales incluidos en la imagen. Puede usar Dockerfiles para crear

imágenes secundarias basadas en las imágenes principales del compilador proporcionadas por Red Hat.

- Si ninguna de las imágenes estándares del compilador S2I proporcionadas por Red Hat satisfacen las necesidades de su aplicación, puede compilar su propia imagen de compilador S2I personalizada.

Imágenes de compilador S2I

Una imagen de compilador S2I es una forma especializada de imagen de contenedor que produce imágenes de contenedor de aplicaciones como salida. Las imágenes de compilador incluyen bibliotecas del sistema operativo base, tiempos de ejecución de lenguaje, marcos (frameworks) y bibliotecas de las que depende una aplicación, y utilidades y herramientas de fuente a imagen.

Por ejemplo, si tiene una aplicación escrita en PHP que desea implementar en OpenShift, puede usar una imagen de compilador PHP para producir una imagen de contenedor de aplicaciones. Puede proporcionar la ubicación del repositorio Git donde el código fuente de la aplicación esté almacenado, y OpenShift combina el código fuente y la imagen de compilador base para producir una imagen de contenedor que OpenShift puede implementar en el clúster. La imagen de contenedor de aplicaciones resultante incluye una versión de Red Hat Enterprise Linux, un tiempo de ejecución PHP y la aplicación. Las imágenes de compilador son un mecanismo conveniente para transformar el código en un contenedor ejecutable de manera rápida y fácil sin la necesidad de crear Dockerfiles.

Uso de imágenes de contenedor

Aunque las compilaciones de fuente a imagen son la forma preferida de compilar e implementar aplicaciones en OpenShift, puede haber escenarios en los que necesite implementar aplicaciones compiladas previamente por terceros. Por ejemplo, algunos proveedores proporcionan imágenes de contenedor totalmente certificadas y soportadas que están listas para ejecutarse. En dichos casos, OpenShift admite implementaciones de imágenes de contenedor compiladas previamente.

El comando `oc new-app` proporciona diferentes maneras flexibles de implementar imágenes de contenedor en un clúster OpenShift. El enfoque más directo es hacer que la imagen de docker compilada previamente esté disponible a través de un registro público (como docker.io o quay.io) o privado (alojado dentro de su organización) y, luego, proporcionar la ubicación de esta imagen al comando `oc new-app`. A continuación, OpenShift extrae la imagen y la implementa en un clúster OpenShift, como cualquier otra imagen compilada en OpenShift.



nota

Puede encontrar un tutorial de ejemplo en el que se demuestran implementaciones de una imagen de contenedor compilada previamente en un clúster OpenShift en Implementaciones binarias de OpenShift [<https://blog.openshift.com/binary-deployments Openshift-3/>]

Red Hat Container Catalog es una fuente de confianza de imágenes de contenedor actualizadas, certificadas y seguras. Contiene imágenes de contenedor simples, así como imágenes de compilador S2I. Las aplicaciones de misión crítica requieren contenedores de confianza. Red Hat compila las imágenes de contenedor de Container Catalog a partir de recursos RPM que han sido analizados por el equipo de seguridad interno de Red Hat y reforzado contra fallas de seguridad.

El portal de Red Hat Container Catalog en <https://registry.redhat.io> proporciona información sobre una serie de imágenes de contenedor que Red Hat basa en versiones de Red Hat Enterprise Linux (RHEL) y sistemas relacionados. Proporciona una serie de imágenes de contenedor listas para usar para empezar a desarrollar aplicaciones para OpenShift.

Red Hat usa un *Índice de estados del contenedor* para la evaluación de los riesgos de seguridad de las imágenes de contenedor que Red Hat proporciona a través de Red Hat Container Catalog. Para obtener más información sobre el sistema de calificación usado por Red Hat en el Índice de estados del contenedor, consulte <https://access.redhat.com/articles/2803031>.

Para obtener más información sobre Red Hat Container Catalog, consulte Preguntas frecuentes (FAQ) [<https://access.redhat.com/containers/#/faq>]

Selección de imágenes de contenedor para aplicaciones

Seleccionar una imagen de contenedor para su aplicación depende de una variedad de factores. Si desea compilar un contenedor personalizado, comience con una imagen de sistema operativo de base (como `rhel7`). Para compilar y ejecutar aplicaciones que requieren herramientas de desarrollo y librerías de tiempo de ejecución específicas, Red Hat proporciona imágenes de contenedor con herramientas como Node.js (`rhscl/nodejs-8-rhel7`), Ruby on Rails (`rhscl/ror-50-rhel7`) y Python (`rhscl/python-36-rhel7`).

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 y que generalmente no se adaptan al calendario de versiones estándar de Red Hat Enterprise Linux (RHEL). Red Hat conserva como parte de RHSCl muchas de las imágenes de contenedor que se ofrecen a través de Red Hat Container Catalog.

Red Hat también proporciona Red Hat OpenShift Application Runtimes (RHOAR), que es la plataforma de desarrollo de Red Hat para aplicaciones nativas de la nube y de microservicios. RHOAR proporciona un enfoque optimizado y soportado con Red Hat para desarrollar aplicaciones de microservicios destinadas a OpenShift como plataforma de implementación.

RHOAR admite varios tiempos de ejecución, lenguajes, marcos (frameworks) y arquitecturas. Ofrece las opciones y la flexibilidad para elegir los marcos (frameworks) y tiempos de ejecución adecuados para el trabajo adecuado. Las aplicaciones desarrolladas con RHOAR pueden ejecutarse en cualquier infraestructura donde se pueda ejecutar Red Hat OpenShift Container Platform, lo que permite evitar la dependencia de un solo proveedor.

RHOAR proporciona:

- Acceso a archivos binarios soportados y creados por Red Hat para marcos (frameworks) de desarrollo y tiempos de ejecución de microservicios seleccionados.
- Acceso a archivos binarios soportados y creados por Red Hat para módulos de integración que reemplazan o mejoran la implementación de un marco (framework) de un patrón de microservicios para usar funciones de OpenShift.
- Compatibilidad para que los desarrolladores puedan escribir aplicaciones con los marcos (frameworks) de desarrollo de microservicios, los tiempos de ejecución y los módulos de integración seleccionados, y con las integraciones con los servicios externos seleccionados, como servidores de bases de datos.
- Compatibilidad de producción para implementar aplicaciones con los marcos (frameworks) de desarrollo de microservicios, los tiempos de ejecución, los módulos de integración y las integraciones seleccionados en un clúster OpenShift soportado.

Creación de imágenes de compilador S2I

Si desea que las aplicaciones usen una imagen de compilador S2I personalizada con su propio conjunto personalizado de tiempos de ejecución, scripts, marcos (frameworks) y librerías, puede crear su propia imagen de compilador S2I. Hay varias opciones para crear imágenes de compilador S2I:

- Cree su propia imagen de compilador S2I desde cero. En escenarios donde su aplicación no puede usar las imágenes de compilador S2I proporcionadas por Container Catalog tal como están, puede compilar una imagen de compilador S2I que personalice el proceso de compilación para satisfacer las necesidades de su aplicación.

OpenShift proporciona la herramienta de línea de comandos `s2i` que lo ayuda a iniciar el entorno de compilación para crear imágenes de compilador S2I personalizadas. Está disponible en el paquete de *fuente a imagen* de los repositorios de Yum de RHSCL (`rhel-server-rhscl-7-rpms`).

- Desvíe una imagen de compilador S2I existente. En lugar de empezar de cero, puede usar los Dockerfiles para las imágenes de compilador existentes en Container Catalog, que están disponibles en <https://github.com/sclorg/?q=s2i>, y personalizarlas conforme a sus necesidades.
- Extienda una imagen de compilador S2I existente. También puede extender una imagen de compilador existente al crear una imagen secundaria y, luego, agregar o reemplazar contenido de la imagen de compilador existente.

Puede encontrar un buen tutorial que lo guía durante el desarrollo de una imagen de compilador S2I personalizada en <https://blog.openshift.com/create-s2i-builder-image/>.



Referencias

Red Hat Container Catalog

<https://access.redhat.com/containers>

Puede encontrar Dockerfiles para imágenes que forman parte de la librería de Red Hat Software Collections en

<https://github.com/sclorg?q=-container>

Puede encontrar más información acerca de las imágenes de contenedor soportadas por Red Hat para usar con OpenShift en el capítulo *Creación de imágenes* de la guía *Imágenes para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/images/creating-images

► Cuestionario

Selección de un enfoque de contenerización

Elija las respuestas correctas para las siguientes preguntas:

Una vez que haya finalizado el cuestionario, haga clic en **CHECK** (VERIFICAR). Si quiere volver a intentarlo, haga clic en **RESET** (RESTABLECER). Haga clic en **SHOW SOLUTION** (MOSTRAR SOLUCIÓN) para ver todas las respuestas correctas.

- 1. **Se le ha pedido que implemente una aplicación comercial basada en .NET de un tercero en un clúster OpenShift, que el proveedor comprimió como una imagen de contenedor. ¿Cuál de las siguientes opciones usaría para implementar la aplicación?**
- a. Una compilación fuente a imagen.
 - b. Un compilador de fuente a imagen personalizado.
 - c. Colocar la imagen de contenedor en un registro de contenedor privado y, luego, implementar la imagen de contenedor en un clúster OpenShift mediante la herramienta de línea de comandos oc de OpenShift.
 - d. Ninguna de estas opciones. No puede implementar aplicaciones basadas en .NET en un clúster OpenShift.
- 2. **Se le pide que implemente una aplicación C++ personalizada, basada en RHEL 7 y desarrollada internamente en un clúster OpenShift. Se le proporcionará el código fuente completo de la aplicación. ¿Cuáles dos de las siguientes opciones se pueden usar, según las prácticas recomendadas, para compilar una imagen de contenedor a fin de implementarla en un clúster OpenShift? (Elija dos opciones).**
- a. Crear un Dockerfile con la imagen de contenedor rhe17 de Red Hat Container Catalog como la base para la aplicación y proporcionarlo a OpenShift mediante un repositorio Git. OpenShift puede crear una imagen de contenedor desde el Dockerfile proporcionado.
 - b. Descargar una imagen de contenedor basada en CentOS 7/C++ desde un registro público, como docker.io, y crear un Dockerfile que compile y comprima la aplicación. Dado que los binarios basados en CentOS 7 son compatibles con RHEL 7, la aplicación funcionará correctamente cuando la implemente en un clúster OpenShift.
 - c. Crear un Dockerfile mediante la imagen de contenedor RHEL 7 de Red Hat Container Catalog como la base y, luego, compilar una imagen de contenedor basada en el Dockerfile. Implementar la imagen de contenedor en un clúster OpenShift mediante la herramienta de línea de comandos oc de OpenShift.
 - d. Crear un Dockerfile mediante cualquier imagen de contenedor basada en Linux de un registro público como base y, luego, compilar una imagen de contenedor basada en el Dockerfile. Implementar la imagen de contenedor en un clúster OpenShift mediante la herramienta de línea de comandos oc de OpenShift.

► **3. Se le ha pedido que migre e implemente dos aplicaciones en un clúster OpenShift en Ruby on Rails y Node.js, respectivamente. Estas aplicaciones se ejecutaron previamente en un entorno con máquinas virtuales. Se le ha proporcionado la ubicación de los repositorios Git donde está ubicado el código fuente de la aplicación para ambas aplicaciones. ¿Cuál de las siguientes opciones se recomienda para implementar las aplicaciones en un clúster OpenShift, teniendo en cuenta que se le ha pedido que mejore estas aplicaciones con más funciones y que continúe desarrollándolas en un entorno de OpenShift?**

- a. Usar las imágenes de contenedor de Ruby on Rails y Node.js de docker.io como base. Crear Dockerfiles personalizados para cada una de estas dos aplicaciones y compilar imágenes de contenedor a partir de ellos. Implementar las imágenes en un clúster OpenShift mediante el proceso de implementación binario estándar.
- b. Crear imágenes de compilador S2I personalizadas, ya que no hay imágenes de compilador disponibles para Ruby on Rails y Node.js en OpenShift.
- c. Usar las imágenes de compilador Ruby on Rails y Node.js S2I de Red Hat Container Catalog, e implementar las aplicaciones en un clúster OpenShift mediante un proceso de compilación S2I estándar.
- d. Usar las imágenes de contenedor basadas en Ruby y Node.js de Red Hat Container Catalog y crear Dockerfiles personalizados para cada una de ellas. Compilar e implementar las imágenes de contenedor resultantes en un clúster OpenShift.

- 4. Se le ha pedido que implemente una aplicación web escrita en el lenguaje de programación Go en un clúster OpenShift. El equipo de seguridad de su organización ha exigido que todas las aplicaciones se ejecuten a través del sistema de análisis de código fuente estático, y un conjunto de pruebas de integración y unidad automatizadas antes de realizar implementaciones en los entornos de producción. El equipo de seguridad también ha proporcionado un Dockerfile personalizado que asegura que todas las aplicaciones se implementen en un sistema operativo basado en RHEL 7 (que está basado en la imagen estándar de RHEL 7 de Red Hat Container Catalog. Su entorno incluye una lista seleccionada de paquetes, usuarios y configuración personalizada para los servicios principales (core) del sistema operativo. Además, el arquitecto de la aplicación ha insistido en que haya una separación clara entre los cambios en el nivel del código fuente y los cambios en la infraestructura (sistema operativo, compilador Go y herramientas Go). Los cambios y parches del sistema operativo o las capas de tiempo de ejecución Go deben desencadenar automáticamente una recompilación y reimplementación de la aplicación.
- ¿Cuál de las siguientes opciones usaría para lograr este objetivo?
- Crear un Dockerfile personalizado que compile una imagen de contenedor de aplicaciones compuesto de la base del sistema operativo RHEL 7, el tiempo de ejecución Go y la herramienta de análisis. Implementar la imagen resultante en un clúster OpenShift mediante el proceso de implementación binario.
 - Crear Dockerfiles separados para la base del sistema operativo RHEL 7, el tiempo de ejecución Go y la herramienta de análisis. OpenShift puede combinar automáticamente los Dockerfiles para producir una única imagen de contenedor de aplicaciones ejecutable.
 - Crear una imagen de compilador S2I personalizada para esta aplicación que incorpore la herramienta de análisis estática, el tiempo de ejecución y compilador Go y la imagen del sistema operativo RHEL 7 basada en el Dockerfile.
 - Crear imágenes de contenedor separadas para la base del sistema operativo RHEL 7, el tiempo de ejecución Go y la herramienta de análisis. Después de que se hayan montado estas imágenes en un registro de imagen de contenedor privado o público, OpenShift puede concatenar automáticamente las capas de cada imagen individual para crear la imagen de contenedor de aplicaciones ejecutable final.
 - Ninguna de estas opciones. Este requisito no se puede cumplir y la aplicación no se puede implementar en un clúster OpenShift.

► Solución

Selección de un enfoque de contenerización

Elija las respuestas correctas para las siguientes preguntas:

Una vez que haya finalizado el cuestionario, haga clic en **CHECK** (VERIFICAR). Si quiere volver a intentarlo, haga clic en **RESET** (RESTABLECER). Haga clic en **SHOW SOLUTION** (MOSTRAR SOLUCIÓN) para ver todas las respuestas correctas.

- 1. **Se le ha pedido que implemente una aplicación comercial basada en .NET de un tercero en un clúster OpenShift, que el proveedor comprimió como una imagen de contenedor. ¿Cuál de las siguientes opciones usaría para implementar la aplicación?**
- a. Una compilación fuente a imagen.
 - b. Un compilador de fuente a imagen personalizado.
 - c. Colocar la imagen de contenedor en un registro de contenedor privado y, luego, implementar la imagen de contenedor en un clúster OpenShift mediante la herramienta de línea de comandos oc de OpenShift.
 - d. Ninguna de estas opciones. No puede implementar aplicaciones basadas en .NET en un clúster OpenShift.
- 2. **Se le pide que implemente una aplicación C++ personalizada, basada en RHEL 7 y desarrollada internamente en un clúster OpenShift. Se le proporcionará el código fuente completo de la aplicación. ¿Cuáles dos de las siguientes opciones se pueden usar, según las prácticas recomendadas, para compilar una imagen de contenedor a fin de implementarla en un clúster OpenShift? (Elija dos opciones).**
- a. Crear un Dockerfile con la imagen de contenedor rhe17 de Red Hat Container Catalog como la base para la aplicación y proporcionarlo a OpenShift mediante un repositorio Git. OpenShift puede crear una imagen de contenedor desde el Dockerfile proporcionado.
 - b. Descargar una imagen de contenedor basada en CentOS 7/C++ desde un registro público, como docker.io, y crear un Dockerfile que compile y comprima la aplicación. Dado que los binarios basados en CentOS 7 son compatibles con RHEL 7, la aplicación funcionará correctamente cuando la implemente en un clúster OpenShift.
 - c. Crear un Dockerfile mediante la imagen de contenedor RHEL 7 de Red Hat Container Catalog como la base y, luego, compilar una imagen de contenedor basada en el Dockerfile. Implementar la imagen de contenedor en un clúster OpenShift mediante la herramienta de línea de comandos oc de OpenShift.
 - d. Crear un Dockerfile mediante cualquier imagen de contenedor basada en Linux de un registro público como base y, luego, compilar una imagen de contenedor basada en el Dockerfile. Implementar la imagen de contenedor en un clúster OpenShift mediante la herramienta de línea de comandos oc de OpenShift.

► **3. Se le ha pedido que migre e implemente dos aplicaciones en un clúster OpenShift en Ruby on Rails y Node.js, respectivamente. Estas aplicaciones se ejecutaron previamente en un entorno con máquinas virtuales. Se le ha proporcionado la ubicación de los repositorios Git donde está ubicado el código fuente de la aplicación para ambas aplicaciones. ¿Cuál de las siguientes opciones se recomienda para implementar las aplicaciones en un clúster OpenShift, teniendo en cuenta que se le ha pedido que mejore estas aplicaciones con más funciones y que continúe desarrollándolas en un entorno de OpenShift?**

- a. Usar las imágenes de contenedor de Ruby on Rails y Node.js de docker.io como base. Crear Dockerfiles personalizados para cada una de estas dos aplicaciones y compilar imágenes de contenedor a partir de ellos. Implementar las imágenes en un clúster OpenShift mediante el proceso de implementación binario estándar.
- b. Crear imágenes de compilador S2I personalizadas, ya que no hay imágenes de compilador disponibles para Ruby on Rails y Node.js en OpenShift.
- c. Usar las imágenes de compilador Ruby on Rails y Node.js S2I de Red Hat Container Catalog, e implementar las aplicaciones en un clúster OpenShift mediante un proceso de compilación S2I estándar.
- d. Usar las imágenes de contenedor basadas en Ruby y Node.js de Red Hat Container Catalog y crear Dockerfiles personalizados para cada una de ellas. Compilar e implementar las imágenes de contenedor resultantes en un clúster OpenShift.

- 4. Se le ha pedido que implemente una aplicación web escrita en el lenguaje de programación Go en un clúster OpenShift. El equipo de seguridad de su organización ha exigido que todas las aplicaciones se ejecuten a través del sistema de análisis de código fuente estático, y un conjunto de pruebas de integración y unidad automatizadas antes de realizar implementaciones en los entornos de producción. El equipo de seguridad también ha proporcionado un Dockerfile personalizado que asegura que todas las aplicaciones se implementen en un sistema operativo basado en RHEL 7 (que está basado en la imagen estándar de RHEL 7 de Red Hat Container Catalog. Su entorno incluye una lista seleccionada de paquetes, usuarios y configuración personalizada para los servicios principales (core) del sistema operativo. Además, el arquitecto de la aplicación ha insistido en que haya una separación clara entre los cambios en el nivel del código fuente y los cambios en la infraestructura (sistema operativo, compilador Go y herramientas Go). Los cambios y parches del sistema operativo o las capas de tiempo de ejecución Go deben desencadenar automáticamente una recompilación y reimplementación de la aplicación.
- ¿Cuál de las siguientes opciones usaría para lograr este objetivo?
- Crear un Dockerfile personalizado que compile una imagen de contenedor de aplicaciones compuesto de la base del sistema operativo RHEL 7, el tiempo de ejecución Go y la herramienta de análisis. Implementar la imagen resultante en un clúster OpenShift mediante el proceso de implementación binario.
 - Crear Dockerfiles separados para la base del sistema operativo RHEL 7, el tiempo de ejecución Go y la herramienta de análisis. OpenShift puede combinar automáticamente los Dockerfiles para producir una única imagen de contenedor de aplicaciones ejecutable.
 - Crear una imagen de compilador S2I personalizada para esta aplicación que incorpore la herramienta de análisis estática, el tiempo de ejecución y compilador Go y la imagen del sistema operativo RHEL 7 basada en el Dockerfile.
 - Crear imágenes de contenedor separadas para la base del sistema operativo RHEL 7, el tiempo de ejecución Go y la herramienta de análisis. Después de que se hayan montado estas imágenes en un registro de imagen de contenedor privado o público, OpenShift puede concatenar automáticamente las capas de cada imagen individual para crear la imagen de contenedor de aplicaciones ejecutable final.
 - Ninguna de estas opciones. Este requisito no se puede cumplir y la aplicación no se puede implementar en un clúster OpenShift.

Compilación de imágenes de contenedor con instrucciones avanzadas de Dockerfile

Objetivos

Tras finalizar esta sección, usted deberá ser capaz de realizar lo siguiente:

- Crear aplicaciones contenedorizadas con la imagen base universal Red Hat.
- Compilar una imagen de contenedor con instrucciones avanzadas de Dockerfile.

Presentación de la imagen base universal de Red Hat

La imagen base universal de Red Hat (*UBI*) tiene como objetivo ser una imagen de contenedor base flexible y de alta calidad para crear aplicaciones contenedorizadas. El objetivo de la imagen base universal Red Hat es permitir que los usuarios creen e implementen aplicaciones en contenedores mediante una imagen base de contenedor de nivel empresarial altamente compatible que sea liviana y eficiente. Puede ejecutar contenedores creados con la imagen base universal en plataformas de Red Hat, así como en plataformas que no sean de Red Hat.

Red Hat deriva la imagen base universal de Red Hat de Red Hat Enterprise Linux (RHEL). Se diferencia de las imágenes base de RHEL 7 existentes, sobre todo por el hecho de que se puede redistribuir bajo los términos del Acuerdo de licencia de usuario final (EULA) de imagen base universal de Red Hat, que permite que los socios, clientes y miembros de la comunidad de Red Hat puedan estandarizar el uso de software y herramientas empresariales bien seleccionadas para ofrecer valor a través de la adición de contenido de terceros.

El plan de soporte es que la imagen base universal siga el mismo ciclo de vida y las mismas fechas de soporte que el contenido RHEL subyacente. Cuando se ejecuta en un nodo de RHEL u OpenShift suscrito, sigue la misma política de soporte que el contenido de RHEL subyacente. Red Hat mantiene una imagen base universal para RHEL 7, que se asigna al contenido de RHEL 7, y otra UBI para RHEL 8, que se asigna al contenido de RHEL 8.

Red Hat recomienda usar la imagen base universal como imagen de contenedor base para nuevas aplicaciones. Red Hat se compromete a seguir admitiendo imágenes base de RHEL anteriores durante el ciclo de vida de soporte de la versión de RHEL.

La imagen base universal de Red Hat consta de:

- Un conjunto de tres imágenes base (`ubi`, `ubi-minimal` y `ubi-init`). Estos reflejan lo que se proporciona para la creación de contenedores con imágenes base de RHEL 7.
- Un conjunto de imágenes de tiempo de ejecución de lenguaje (`java`, `php`, `python`, `ruby`, `nodejs`). Estas imágenes de tiempo de ejecución permiten que los desarrolladores empiecen a desarrollar aplicaciones con la confianza que proporciona una imagen de contenedor soportada y creada por Red Hat.
- Un conjunto de repositorios y canales de Yum asociados que incluyen paquetes y actualizaciones de RPM. Estos permiten agregar dependencias de aplicaciones y reconstruir imágenes de contenedor según sea necesario.

Tipos de imágenes base universal

La imagen base universal de Red Hat proporciona tres imágenes base principales:

ubi

Una imagen base estándar basada en paquetes de nivel empresarial de RHEL. Sirve para la mayoría de los casos de uso de aplicaciones.

ubi-minimal

Una imagen base mínima creada con `microdnf`, una versión reducida de la utilidad `dnf`. Esto proporciona la imagen de contenedor más pequeña.

ubi-init

Esta imagen le permite ejecutar fácilmente varios servicios, como servidores web, servidores de aplicaciones y bases de datos, todo en un único contenedor. Le permite usar el conocimiento integrado en los archivos de unidad `systemd` sin tener que determinar cómo iniciar el servicio.

Ventajas de la imagen base universal de Red Hat

El uso de una imagen base universal de Red Hat como imagen base para las aplicaciones contenedorizadas tiene varias ventajas:

- **Tamaño mínimo:** La imagen base universal es una imagen de contenedor base relativamente mínima (aproximadamente 90-200 MB) con tiempos de inicio rápidos.
- **Seguridad:** La procedencia es una gran preocupación cuando se usan imágenes de contenedores base. Debe usar una imagen de confianza de una fuente confiable. Los tiempos de ejecución de lenguaje, los servidores web y las librerías principales (core), como OpenSSL, tienen un impacto en la seguridad cuando se trasladan a producción. La imagen base universal recibe actualizaciones de seguridad oportunas de los equipos de seguridad de Red Hat.
- **Rendimiento:** el equipo interno de ingeniería de rendimiento de Red Hat prueba, optimiza y certifica las imágenes base. Se trata de imágenes de contenedor probadas que se usan ampliamente en algunas de las cargas de trabajo más sensibles a las fallas y de uso más intensivo de cálculos y E/S del mundo.
- **Soporte para socios, ISV y certificación de proveedores:** La imagen base universal hereda el amplio ecosistema de socios, ISV y proveedores externos de RHEL que admiten miles de aplicaciones. La imagen base universal facilita a estos socios la creación, implementación y certificación de aplicaciones y les permite implementar la aplicación contenedora resultante tanto en plataformas de Red Hat, como RHEL y OpenShift, como en plataformas de contenedores que no son de Red Hat.
- **Una misma compilación se puede implementar en muchos hosts diferentes:** La imagen base universal de Red Hat se puede compilar e implementar en cualquier sistema: en OpenShift/RHEL o en cualquier otro host de contenedor (Fedora, Debian, Ubuntu y más).

Instrucciones avanzadas de Dockerfile

Un `Dockerfile` automatiza la compilación de imágenes de contenedor. Un `Dockerfile` es un archivo de texto que contiene un conjunto de instrucciones para compilar una imagen de contenedor. Las instrucciones se ejecutan individualmente en orden secuencial. En esta sección se revisan algunas de las instrucciones básicas de `Dockerfile` y, a continuación, se describen algunas instrucciones más avanzadas, incluidas formas prácticas de usarlas para crear imágenes de contenedor para su implementación en OpenShift.

La instrucción RUN

La instrucción RUN ejecuta comandos en una nueva capa sobre la imagen actual; luego, confirma los resultados. El proceso de compilación de contenedores usa el resultado confirmado en el siguiente paso en el Dockerfile. El proceso de compilación de contenedor usa /bin/sh para ejecutar comandos.

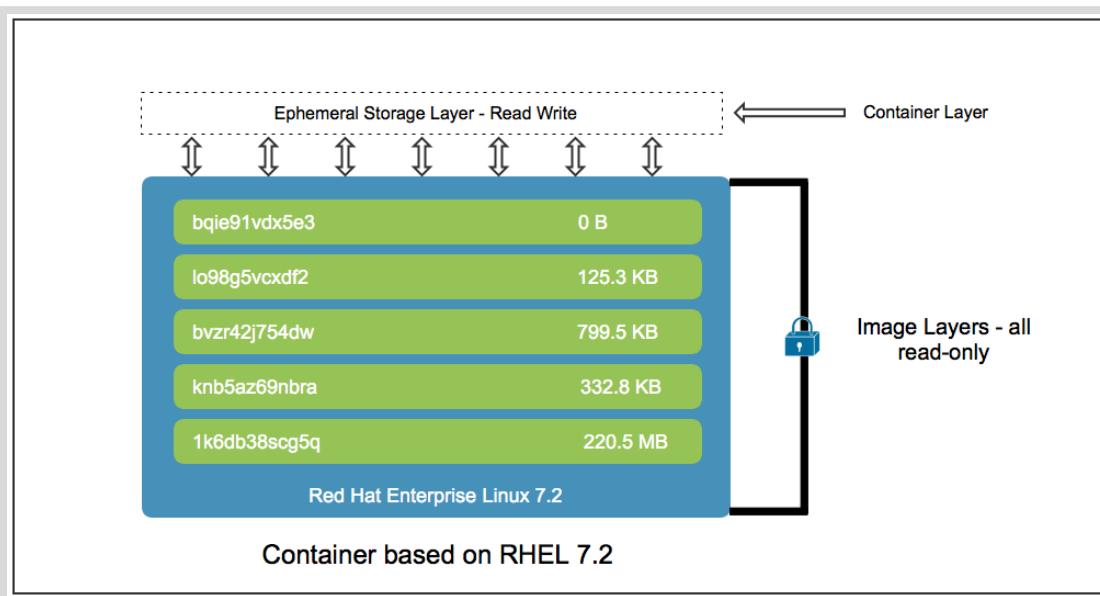


Figura 2.1: Capas de una imagen de contenedor

Cada instrucción de un Dockerfile da como resultado una nueva capa para la imagen de contenedor final. Por lo tanto, tener demasiadas instrucciones en un Dockerfile crea demasiadas capas, lo que genera imágenes demasiado grandes. Por ejemplo, considere la siguiente instrucción RUN en un Dockerfile:

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms"
RUN yum update
RUN yum install -y httpd
RUN yum clean all -y
```

El ejemplo anterior no es una práctica adecuada cuando se crean imágenes de contenedor, ya que se crean cuatro capas con un solo propósito. Al crear Dockerfiles, Red Hat recomienda que minimice siempre la cantidad de capas. Puede lograr el mismo objetivo mediante el separador de comandos && para ejecutar varios comandos dentro de una única instrucción RUN:

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms" && \
    yum update && \
    yum install -y httpd && \
    yum clean all -y
```

El ejemplo actualizado crea solo una capa y la legibilidad no se ve comprometida.

**nota**

En Docker 1.10 y versiones posteriores, solo las instrucciones RUN, COPY y ADD crean capas. Otras instrucciones crean imágenes intermedias temporales y no aumentan directamente el tamaño de la imagen.

La instrucción LABEL

La instrucción LABEL define metadatos de la imagen. Las etiquetas son pares clave-valor que están adjuntos a una imagen. Por lo general, la instrucción LABEL agrega metadatos descriptivos a la imagen, como versiones, una breve descripción y otros detalles que proporcionan información a los usuarios de la imagen.

Al compilar imágenes para OpenShift, use `io.openshift` como prefijo del nombre de la etiqueta para distinguir entre los metadatos relacionados con OpenShift y Kubernetes. Las herramientas de OpenShift pueden analizar ciertas etiquetas y realizar ciertas acciones en función de la presencia de estas etiquetas. En la siguiente tabla, se muestran algunas de las etiquetas más usadas:

Etiquetas soportadas de OpenShift

| Nombre de la etiqueta | Descripción |
|---|--|
| <code>io.openshift.tags</code> | Esta etiqueta contiene una lista de etiquetas separadas por comas. Las etiquetas categorizan las imágenes de contenedor en amplias áreas de funcionalidad. Las etiquetas ayudan a la interfaz de usuario y a las herramientas de generación a sugerir imágenes de contenedor relevantes durante el proceso de creación de aplicaciones. |
| <code>io.k8s.description</code> | Esta etiqueta les proporciona a los consumidores de la imagen de contenedor información más detallada acerca del servicio o de la funcionalidad que esta imagen brinda. |
| <code>io.openshift.expose-services</code> | <p>Esta etiqueta contiene una lista de puertos de servicio que coinciden con las instrucciones EXPOSE en el Dockerfile y brinda información más detallada acerca del servicio que se proporciona a los consumidores.</p> <p>El formato es <code>PORT[/PROTO]:NAME</code> donde <code>[/PROTO]</code> es opcional y usa <code>tcp</code> como valor predeterminado si no está especificado.</p> |

Para ver una lista completa de todas las etiquetas específicas de OpenShift, sus descripciones y usos de ejemplo, consulte las referencias al final de esta sección.

La instrucción WORKDIR

La instrucción WORKDIR determina el directorio de trabajo para cualquier instrucción RUN, CMD, ENTRYPOINT, COPY o ADD en un Dockerfile.

Red Hat recomienda usar rutas absolutas en las instrucciones WORKDIR. Use WORKDIR en lugar de varias instrucciones RUN, donde puede cambiar directorios y, luego, ejecutar algunos comandos. Este enfoque garantiza un mejor mantenimiento a largo plazo y es más fácil para la solución de problemas.

La instrucción ENV

La instrucción ENV define variables de entorno que estarán disponibles en el contenedor. Puede declarar varias instrucciones ENV dentro del Dockerfile. Puede usar el comando env dentro del contenedor para ver cada una de las variables de entorno.

Se recomienda usar las instrucciones ENV para definir rutas de archivo y carpeta en lugar de codificarlas en las instrucciones de Dockerfile. Es útil almacenar información, como números de versión de software, y anexar directorios a la variable de entorno PATH.

El concepto de capas también se aplica a instrucciones, como ENV y LABEL. Para especificar varias instrucciones LABEL o ENV, Red Hat recomienda que use una sola instrucción para todas las etiquetas y variables de entorno, y que separe cada par de clave-valor con un signo igual (=):

```
LABEL version="2.0" \
    description="This is an example container image" \
    creationDate="01-09-2017"
```

```
ENV MYSQL_DATABASE_USER="my_database_user" \
    MYSQL_DATABASE="my_database"
```

La instrucción USER

Red Hat recomienda que ejecute la imagen como un usuario que no es root por motivos de seguridad. Para reducir la cantidad de capas, evite usar la instrucción USER demasiadas veces en un Dockerfile. Más adelante en esta sección hablaremos sobre las implicaciones de seguridad específicas a la ejecución de contenedores como un usuario que no es root.



Advertencia

OpenShift, de manera predeterminada, no respeta la instrucción USER definida por la imagen de contenedor. Por motivos de seguridad, OpenShift usa un ID de usuario aleatorio diferente al ID de usuario root (0) para ejecutar contenedores.

La instrucción VOLUME

La instrucción VOLUME crea un punto de montaje dentro del contenedor e indica a los consumidores de la imagen que los volúmenes montados externamente desde la máquina host u otros contenedores se puede vincular con este punto de montaje.

Red Hat recomienda el uso de las instrucciones VOLUME para datos persistentes. OpenShift puede montar el almacenamiento conectado a la red al nodo que ejecuta el contenedor y, si el contenedor se traslada a un nuevo nodo, el almacenamiento se vuelve a conectar a ese nodo. Al usar el volumen para todas las necesidades de almacenamiento persistente, el contenido se conserva incluso si se reinicia o traslada el contenedor.

Además, definir volúmenes explícitamente en su Dockerfile hace que los consumidores de la imagen comprendan más fácilmente los volúmenes que pueden definir cuando ejecutan su imagen.

Compilación de imágenes con la instrucción ONBUILD

La instrucción `ONBUILD` registra los desencadenadores en la imagen de contenedor. Un Dockerfile usa `ONBUILD` para declarar instrucciones que se ejecutan únicamente cuando compila una imagen secundaria.

La instrucción `ONBUILD` es útil para admitir una personalización simple de una imagen de contenedor para casos de uso comunes, como cargar datos previamente o proporcionar configuración personalizada a una aplicación. En la imagen principal, se proporcionan comandos que son comunes para todas las imágenes secundarias downstream. La imagen secundaria solo brinda los archivos de configuración y datos. El Dockerfile para la imagen secundaria puede ser tan simple como tener únicamente la instrucción `FROM` que hace referencia a la imagen principal.



nota

La instrucción `ONBUILD` no se incluye en la especificación OCI y, por lo tanto, no está soportada de forma predeterminada al crear contenedores con Podman o Buildah. Use la opción `--format docker` para habilitar el soporte para la instrucción `ONBUILD`.

Por ejemplo, supongamos que está compilando una imagen principal Node.js que desea que todos los desarrolladores de su organización usen como base cuando compilen aplicaciones con los siguientes requisitos:

- Aplique ciertos estándares, como copiar fuentes de JavaScript a la carpeta de la aplicación, para que puedan ser interpretados por el motor Node.js.
- Ejecute el comando `npm install`, que busca todas las dependencias descritas en el archivo `package.json`.

No puede incrustar estos requisitos como instrucciones en el Dockerfile principal debido a que no tiene el código fuente de la aplicación, y cada aplicación puede tener diferentes dependencias enumeradas en su archivo `package.json`.

Declare instrucciones `ONBUILD` en el Dockerfile principal. Debajo se muestra el Dockerfile principal:

```
FROM registry.access.redhat.com/rhscl/nodejs-6-rhel7
EXPOSE 3000
# Mandate that all Node.js apps use /usr/src/app as the main folder (APP_ROOT).
RUN mkdir -p /opt/app-root/
WORKDIR /opt/app-root

# Copy the package.json to APP_ROOT
ONBUILD COPY package.json /opt/app-root

# Install the dependencies
ONBUILD RUN npm install

# Copy the app source code to APP_ROOT
ONBUILD COPY src /opt/app-root

# Start node server on port 3000
CMD [ "npm", "start" ]
```

Suponiendo que compila la imagen de contenedor principal y la denomina `mynodejs-base`, un Dockerfile secundario para una aplicación que usa la imagen principal tiene el siguiente aspecto:

```
FROM mynodejs-base
RUN echo "Started Node.js server..."
```

Cuando se inicia el proceso de compilación para la imagen secundaria, desencadena la ejecución de las tres instrucciones `ONBUILD` definidas en la imagen principal, antes de invocar la instrucción `RUN` en el Dockerfile secundario.

Consideraciones de OpenShift para la instrucción USER

De manera predeterminada, OpenShift ejecuta los contenedores usando un ID de usuario asignado de manera arbitraria. Este enfoque mitiga el riesgo de que los procesos que se ejecutan en el contenedor reciban privilegios aumentados en la máquina host debido a vulnerabilidades de seguridad en el motor del contenedor.

Adaptación de Dockerfiles para OpenShift

Cuando escribe o modifica un Dockerfile que compila una imagen para ejecutar en un clúster OpenShift, necesita abordar lo siguiente:

- Los directorios y archivos de los que leen o a los que escriben los procesos en el contenedor deben pertenecer al grupo `root` y tener permisos de lectura o escritura en el grupo.
- Los archivos ejecutables deben contar con permisos de ejecución de grupos.
- Los procesos que se ejecutan en el contenedor no se deben escuchar en puertos privilegiados (es decir, puertos por debajo de 1024), ya que no se ejecutan como usuarios privilegiados.

Si agrega la siguiente instrucción `RUN` a su Dockerfile, se definen los permisos de archivo y directorio para permitir que los usuarios del grupo `root` accedan a ellos en el contenedor:

```
RUN chgrp -R 0 directory && \
    chmod -R g=u directory
```

La cuenta de usuario que ejecuta el contenedor es siempre un miembro del grupo `root`; por lo tanto, el contenedor puede leer y escribir en este directorio. El grupo `root` no tiene permisos especiales (a diferencia del usuario `root`), lo que minimiza los riesgos de seguridad con esta configuración.

El argumento `g=u` del comando `chmod` iguala los permisos del grupo con los permisos del usuario propietario, que de manera predeterminada son de lectura y escritura. Puede usar el argumento `g+rwx` con los mismos resultados.

Ejecución de contenedores como root mediante restricciones del contexto de seguridad (SCC)

En algunos casos, puede no tener acceso a los Dockerfiles para determinadas imágenes. Es posible que deba ejecutar la imagen como el usuario `root`. En este escenario, necesita configurar OpenShift para permitir que los contenedores se ejecuten como `root`.

OpenShift proporciona *restricciones de contexto de seguridad (SCC)* que controlan las acciones que un pod puede realizar y los recursos a los que puede acceder. OpenShift se envía con varias

SCC incorporadas. De manera predeterminada, todos los contenedores creados por OpenShift usan la SCC denominada **restricted**, que ignora el ID de usuario determinado por la imagen de contenedor y asigna un ID de usuario aleatorio a los contenedores.

Para que los contenedores puedan usar un ID de usuario fijo, como 0 (el usuario **root**), necesita usar la SCC **anyuid**. Para ello, primero necesita crear una *cuenta de servicio*. Una cuenta de servicio es la identidad de OpenShift para un pod. Todos los pods de un proyecto se ejecutan en una cuenta de servicio predeterminada, a menos que el pod, o su configuración de implementación, esté configurado de otra manera.

Si tiene una aplicación que requiere una capacidad no otorgada por la SCC restricted, cree una nueva cuenta de servicio específica y agréguela a la SCC correspondiente. A continuación, cambie la configuración de implementación que crea los pods de aplicación para usar la nueva cuenta de servicio.

Los siguientes pasos detallan cómo permitir que los contenedores se ejecuten como el usuario **root** en un proyecto de OpenShift:

- Cree una nueva cuenta de servicio:

```
[user@host ~]$ oc create serviceaccount myserviceaccount
```

- Modifique la configuración de implementación para que la aplicación use la nueva cuenta de servicio. Use el comando `oc patch` para hacer esto:

```
[user@host ~]$ oc patch dc/demo-app --patch \
> '{"spec":{"template":{"spec":{"serviceAccountName": "myserviceaccount"}}}}'
```



nota

Para obtener más información sobre cómo usar el comando `oc patch`, consulte Guía de administradores de OpenShift para `oc patch` [<https://access.redhat.com/articles/3319751>]. Ejecute el comando `oc patch -h` para mostrar el uso.

- Agregue la cuenta de servicio `myserviceaccount` a la SCC `anyuid` para que se ejecute mediante un ID de usuario fijo en el contenedor:

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



Referencias

Prácticas recomendadas para escribir Dockerfiles

https://docs.docker.com/engine/userguide/eng-image/dockerfile_best-practices

Para obtener más información sobre la creación de imágenes, consulte el capítulo *Creación de imágenes* de la guía *Imágenes* en la documentación de productos de OpenShift Container Platform, disponible en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/images/creating-images

► Ejercicio Guiado

Compilación de imágenes de contenedor con instrucciones avanzadas de Dockerfile

En este ejercicio, usará Red Hat OpenShift para compilar e implementar un contenedor de servidor HTTP Apache a partir de un Dockerfile.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear una imagen de contenedor de servidor HTTP Apache con un Dockerfile e implementarla en un clúster OpenShift.
- Crear una imagen del contenedor secundario extendiendo la imagen del servidor HTTP Apache principal.
- Cambiar el Dockerfile por la imagen del contenedor secundario, de manera que se ejecute en un clúster OpenShift con un ID de usuario aleatorio.

Andes De Comenzar

Para realizar este ejercicio, necesita acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen principal del servidor HTTP Apache (quay.io/redhattraining/httpd-parent).
- El Dockerfile para la imagen del contenedor secundario en el repositorio Git (container-build).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

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

- 1. Revise el Dockerfile principal del servidor HTTP Apache.

Se ofrece una imagen compilada previamente de contenedor principal del servidor HTTP Apache en el registro público de Quay.io en `quay.io/redhattraining/httpd-parent`. Revise brevemente el Dockerfile para esta imagen principal ubicada en `~/D0288/labs/container-build/httpd-parent/Dockerfile`:

```
FROM registry.access.redhat.com/ubi8/ubi:8.0 ①  
MAINTAINER Red Hat Training <training@redhat.com>
```

```

# DocumentRoot for Apache
ENV DOCROOT=/var/www/html ②

RUN yum install -y --no-docs --disableplugin=subscription-manager httpd && \
    yum clean all --disableplugin=subscription-manager -y && \
    echo "Hello from the httpd-parent container!" > ${DOCROOT}/index.html ③

# Allows child images to inject their own content into DocumentRoot
ONBUILD COPY src/ ${DOCROOT}/ ④

EXPOSE 80

# This stuff is needed to ensure a clean start
RUN rm -rf /run/httpd && mkdir /run/httpd

# Run as the root user
USER root ⑤

# Launch httpd
CMD /usr/sbin/httpd -DFOREGROUND

```

- ① La imagen base es la imagen base universal (UBI) para Red Hat Enterprise Linux 8.0 de Red Hat Container Catalog.
- ② Las variables del entorno para esta imagen de contenedor.
- ③ La instrucción RUN (Ejecutar) contiene varios comandos que instalan el servidor HTTP Apache y crean una página de inicio predeterminada para el servidor web.
- ④ La instrucción ONBUILD permite que las imágenes secundarias proporcionen su propio contenido de servidor web personalizado al compilar una imagen que deriva de esta imagen principal.
- ⑤ La instrucción USER (Usuario) ejecuta el proceso del servidor HTTP Apache como el usuario root.



nota

Observe cómo las líneas RUN combinan varios comandos en una única instrucción, siempre que sea posible, para reducir la cantidad de capas en la imagen. Esto genera imágenes más pequeñas, que son más rápidas de implementar.

► 2. Revise el Dockerfile secundario del servidor HTTP Apache.

Use la imagen de contenedor del servidor HTTP Apache principal (`redhat-training/httpd-parent`) como base para ampliar y personalizar la imagen a fin de adaptarla a su aplicación. El Dockerfile para la imagen de contenedor secundario se almacena en el servidor de repositorios Git del aula. Para ver el Dockerfile, realice los siguientes pasos:

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. Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación `master` del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 2.3. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b container-build
Switched to a new branch 'container-build'
[student@workstation D0288-apps]$ git push -u origin container-build
...output omitted...
* [new branch]      container-build -> container-build
Branch container-build set up to track remote branch container-build from origin.
```

- 2.4. Inspecione el archivo ~/D0288-apps/container-build/Dockerfile. El Dockerfile tiene una sola instrucción, FROM, que usa la imagen redhattraining/httpd-parent:

```
FROM quay.io/redhattraining/http-parent
```

- 2.5. El contenedor secundario ofrece su propio archivo index.html en la carpeta ~/D0288-apps/container-build/src, el cual sobrescribe el archivo index.html del contenedor principal. A continuación, se muestra el contenido del archivo index.html de la imagen del contenedor secundario:

```
<!DOCTYPE html>
<html>
<body>
  Hello from the Apache child container!
</body>
</html>
```

- 3. Compile e implemente un contenedor en un clúster OpenShift con el Dockerfile secundario del servidor HTTP Apache.

- 3.1. Inicie sesión en OpenShift con su nombre de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

- 3.2. Cree un nuevo proyecto para la aplicación. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador.

```
[student@workstation D0288-apps]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-container-build
Now using project "youruser-container-build" on server "https://
api.cluster.domain.example.com:6443".
```

3.3. Compile e implemente la imagen secundaria del servidor HTTP Apache:

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config --name hola \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#container-build \
> --context-dir container-build
--> Found Docker image 1d6f8d3 (11 minutes old) from quay.io for "quay.io/
redhattraining/httpd-parent"

Red Hat Universal Base Image 8

-----
The Universal Base Image is designed and engineered to be the base layer for
all of your containerized applications, middleware and utilities. This base image
is freely redistributable, but Red Hat only supports Red Hat technologies through
subscriptions for Red Hat products. This image is maintained by Red Hat and
updated regularly.

Tags: base rhel8

...output omitted...
--> Success
Build scheduled, use 'oc logs -f bc/hola' 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/hola'
Run 'oc status' to view your app.
...output omitted...
```

▶ 4. Vea los registros de compilación.

**Importante**

El proceso de compilación puede tardar un tiempo en iniciarse. Si ve la siguiente salida, vuelva a ejecutar el comando.

```
Error from server (BadRequest): unable to wait for build hola-1 to run: timed
out waiting for the condition
```

```
[student@workstation D0288-apps]$ oc logs -f bc/hola
Cloning "https://github.com/youruser/D0288-apps" ... ①
Commit: 823cd7a7476b5664cb267e5d0ac611de35df9f07 (Initial commit)
Author: Your Name <youremail@example.com>
Date: Sun Jun 9 20:45:53 2019 -0400
Replaced Dockerfile FROM image quay.io/redhattraining/httpd-parent
Caching blobs under "/var/cache/blobs".

Pulling image quay.io/redhattraining/httpd-parent@sha256:3e454fdac5 ②
...output omitted...
Getting image source signatures
Copying blob sha256:d02c3bd
...output omitted...
Writing manifest to image destination
```

```

Storing signatures
STEP 1: FROM quay.io/redhattraining/httpd-parent@sha256:2833...86ff
STEP 2: COPY src/ ${DOCROOT}/3
...output omitted...
Successfully pushed //image-registry.openshift-image-registry.svc:5000/... 4
Push successful

```

- 1** OpenShift clona el repositorio Git de la URL provista por el comando `oc new-app`.
- 2** El Dockerfile en la raíz del repositorio Git se identifica automáticamente y se inicia un proceso de compilación de Docker.
- 3** La instrucción `ONBUILD` del Dockerfile principal desencadena la copia del archivo `index.html` del contenedor secundario, la cual sobrescribe el archivo del índice principal.
- 4** Por último, la imagen compilada se inserta (push) en el registro interno OpenShift.

- ▶ 5. Verifique que el pod de la aplicación no logra iniciarse. El pod se encontrará en estado de `Error`, pero si espera demasiado, el pod pasará al estado `CrashLoopBackOff`.

```
[student@workstation D0288-apps]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
hola-1-build 0/1     Completed  0          22s
hola-1-deploy 1/1     Running   0          14s
hola-1-p75f5 0/1     Error    0          12s
```

- ▶ 6. Inspeccione los registros del contenedor para ver el motivo por el que el pod no logró iniciarse:

```
[student@workstation D0288-apps]$ oc logs hola-1-p75f5
AH00558: httpd: Could not reliably determine the server's fully qualified domain
name...
(13)Permission denied: AH00072: make_sock: could not bind to address [::]:80 1
(13)Permission denied: AH00072: make_sock: could not bind to address 0.0.0.0:80 2
no listening sockets available, shutting down
AH00015: Unable to open logs 3
```

- 1 2** Debido a que OpenShift ejecuta contenedores con un ID de usuario aleatorio, los puertos inferiores a 1024 cuentan con privilegios y solo se pueden ejecutar como root.
- 3** El ID de usuario aleatorio que usa OpenShift para ejecutar el contenedor no tiene permisos de lectura ni de escritura de archivos de registro en `/var/log/httpd` (la ubicación predeterminada del archivo de registro para el servidor HTTP Apache en RHEL 7).



Advertencia

El pod de la aplicación con error se elimina después de un breve tiempo. Asegúrese de inspeccionar los archivos de registro del pod de la aplicación antes de que el pod se elimine.

- 7. Elimine todos los recursos del proyecto de OpenShift. Como próximo paso, cambie el Dockerfile para seguir las recomendaciones de Red Hat para OpenShift.

Antes de actualizar el Dockerfile del servidor HTTP Apache secundario, elimine todos los recursos en el proyecto que fueron creados hasta el momento:

```
[student@workstation D0288-apps]$ oc delete all -l app=hola
pod "hola-2-gbx6f" deleted
replicationcontroller "hola-1" deleted
service "hola" deleted
deploymentconfig.apps.openshift.io "hola" deleted
buildconfig.build.openshift.io "hola" deleted
build.build.openshift.io "hola-1" deleted
imagestream.image.openshift.io "httpd-parent" deleted
imagestream.image.openshift.io "hola" deleted
```

- 8. Cambie el Dockerfile para que el contenedor secundario se ejecute en un clúster OpenShift mediante la actualización del proceso del servidor HTTP Apache para que se ejecute como usuario aleatorio sin privilegios.
- 8.1. Edite el archivo ~/D0288-apps/container-build/Dockerfile y realice los siguientes pasos. También puede copiar estas instrucciones del archivo ~/D0288/solutions/container-build/Dockerfile provisto.
 - 8.2. Sobrescriba la instrucción EXPOSE de la imagen principal y cambie el puerto a 8080.

EXPOSE 8080

- 8.3. Incluya la etiqueta io.openshift.expose-service para indicar el puerto cambiado en el que se ejecuta el servidor web:

LABEL io.openshift.expose-services="8080:http"

Actualice la lista de etiquetas para incluir las etiquetas io.k8s.description, io.k8s.display-name e io.openshift.tags que OpenShift consume para proporcionar metadatos útiles sobre la imagen del contenedor:

```
LABEL io.k8s.description="A basic Apache HTTP Server child image, uses ONBUILD" \
io.k8s.display-name="Apache HTTP Server" \
io.openshift.expose-services="8080:http" \
io.openshift.tags="apache, httpd"
```

- 8.4. Debe ejecutar el servidor web en un puerto sin privilegios (es decir, mayor que 1024). Use una instrucción RUN para cambiar el número de puerto del archivo de configuración del servidor HTTP Apache del puerto predeterminado 80 a 8080:

RUN sed -i "s/Listen 80/Listen 8080/g" /etc/httpd/conf/httpd.conf

- 8.5. Cambie el ID de grupo y los permisos de las carpetas en las que el proceso del servidor web lee y escribe archivos:

```
RUN chgrp -R 0 /var/log/httpd /var/run/httpd && \
    chmod -R g=u /var/log/httpd /var/run/httpd
```

- 8.6. Agregue una instrucción USER para un usuario sin privilegios. La convención de Red Hat es usar el ID de usuario 1001:

USER 1001

- 8.7. Guarde el Dockerfile y confirme los cambios al repositorio Git de la carpeta ~/D0288-apps/container-build:

```
[student@workstation D0288-apps]$ cd container-build
[student@workstation container-build]$ git commit -a -m \
> "Changed Dockerfile to enable running as a random uid on OpenShift"
...output omitted...
[student@workstation container-build]$ git push
...output omitted...
[student@workstation container-build]$ cd ~
```

- 9. Vuelva a compilar y a implementar la imagen del contenedor secundario del servidor HTTP Apache.

- 9.1. Recree la aplicación usando el nuevo Dockerfile:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name hola \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#container-build \
> --context-dir container-build
```

- 9.2. Espere hasta que finalice la compilación, y el pod esté listo y en ejecución. Consulte el estado del pod de la aplicación:

```
[student@workstation ~]$ oc get pods
NAME        READY     STATUS    RESTARTS   AGE
hola-1-75gkw 1/1      Running   0          5s
hola-1-build  0/1      Completed  0          15s
```

El pod de la aplicación ahora se iniciará correctamente y aparecerá en estado Running (En ejecución).

- 10. Cree una ruta OpenShift para exponer la aplicación a acceso externo:

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

- 11. Obtenga la URL de enrutamiento con el comando oc get route:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT                               PATH      SERVICES
PORT      TERMINATION    WILDCARD
hola      hola-youruser-container-build.cluster.domain.example.com   hola
8080-tcp           None
```

- 12. Pruebe la aplicación con la URL de ruta que obtuvo en el paso anterior:

```
[student@workstation ~]$ curl \
> http://hola-${RHT_OCP4_DEV_USER}-container-build.${RHT_OCP4_WILDCARD_DOMAIN}
...output omitted...
Hello from the Apache child container!
...output omitted...
```

- 13. Realice la limpieza. Elimine el proyecto:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-container-build
```

Finalizar

En workstation, ejecute el comando `lab container-build finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

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

Esto concluye el ejercicio guiado.

Inserción de datos de configuración en una aplicación

Objetivos

Tras completar esta sección, deberá ser capaz de seleccionar un método para insertar datos de configuración en una aplicación y crear los recursos necesarios para hacerlo.

Externalización de la configuración de aplicaciones en OpenShift

Por lo general, los desarrolladores configuran las aplicaciones a través de una combinación de variables del entorno, argumentos de la línea de comandos y archivos de configuración. Cuando implementa aplicaciones en OpenShift, la gestión de configuración representa un desafío debido a la naturaleza inmutable de los contenedores. A diferencia de las implementaciones tradicionales que no están contenedorizadas, no se recomienda asociar la aplicación con la configuración al ejecutar aplicaciones contenedorizadas.

El enfoque recomendado para las aplicaciones en contenedores es desasociar los binarios estáticos de la aplicación de los datos dinámicos de configuración y externalizar la configuración. Esta separación garantiza la movilidad de las aplicaciones en muchos entornos.

Por ejemplo, supongamos que desea mover una aplicación que está implementada en un clúster OpenShift de un entorno de desarrollo a un entorno de producción, con etapas intermedias como pruebas y aceptación del usuario. Debe usar la misma imagen de contenedor de aplicaciones en todas las etapas y tener los detalles de configuración específicos de cada entorno fuera de la imagen de contenedor.

Uso de recursos de secretos y mapa de configuración

OpenShift proporciona tipos de recursos de secretos y mapa de configuración para externalizar y administrar la configuración de las aplicaciones.

Se usan recursos de secretos para almacenar información confidencial, como contraseñas, claves y tokens. Como desarrollador, es importante crear secretos para evitar poner en peligro las credenciales y otra información confidencial de la aplicación. Hay diferentes tipos de secretos que se pueden usar para aplicar nombres de usuario y claves en el objeto secreto: `service-account-token`, `basic-auth`, `ssh-auth`, `tls` y `opaque`. El tipo predeterminado es `opaque`. El tipo `opaque` no realiza ninguna validación y permite pares no estructurados de clave:valor que pueden contener valores arbitrarios.

Los recursos del mapa de configuración son similares a los recursos secretos, aunque almacenan datos no confidenciales. Se puede usar un recurso de mapa de configuración para almacenar información de granularidad fina, como propiedades individuales, o información de granularidad gruesa, como archivos de configuración completos o datos de JSON.

Puede crear recursos de secretos y mapa de configuración mediante la consola web o CLI de OpenShift. A continuación, puede hacer referencia a ellos en la especificación de su pod y OpenShift inserta automáticamente los datos del recurso en el contenedor como variables de entorno, o como archivos montados a través de volúmenes dentro del contenedor de aplicaciones.

También puede cambiar la configuración de implementación de una aplicación en ejecución para hacer referencia a recursos de secretos y mapa de configuración. A continuación, OpenShift vuelve a implementar automáticamente la aplicación y pone los datos a disposición del contenedor.

Los datos se almacenan dentro de un recurso secreto usando la codificación base64. Cuando se inyectan datos de un secreto en un contenedor, estos se decodifican y se montan como un archivo o se inyectan como variables de entorno dentro del contenedor.

Características de los secretos y mapas de configuración

Tenga en cuenta lo siguiente con respecto a los secretos y mapas de configuración:

- Se pueden citar independientemente de su definición.
- Por motivos de seguridad, los volúmenes montados para estos recursos están respaldados por instalaciones de almacenamiento de archivos temporales (tmpfs) y nunca se almacenan en un nodo.
- Se incluyen en un espacio de nombres.

Creación y administración de secretos y mapas de configuración

Los secretos y mapas de configuración deben crearse antes de crear los pods que dependen de ellos. Puede usar la CLI o la consola web para crear estos recursos.

Uso de la línea de comandos

Use el comando `oc create` para crear recursos de secretos y mapa de configuración.

Para crear un nuevo mapa de configuración que almacene literales de cadenas:

```
[user@host ~]$ oc create configmap config_map_name \
> --from-literal key1=value1 \
> --from-literal key2=value2
```

Para crear un nuevo secreto que almacene literales de cadenas:

```
[user@host ~]$ oc create secret generic secret_name \
> --from-literal username=user1 \
> --from-literal password=mypa55w0rd
```

Para crear un nuevo mapa de configuración que almacene el contenido de un archivo o directorio que contiene un conjunto de archivos:

```
[user@host ~]$ oc create configmap config_map_name \
> --from-file /home/demo/conf.txt
```

Cuando crea un mapa de configuración desde un archivo, de manera predeterminada, el nombre de la clave será el nombre del archivo, y el valor será el contenido del archivo.

Cuando crea un recurso de mapa de configuración basado en un directorio, cada archivo cuyo nombre es una clave válida en el directorio se almacena en el mapa de configuración. Los subdirectorios, enlaces simbólicos, archivos de dispositivo y tuberías se ignoran.

Ejecute el comando `oc create configmap --help` para obtener más información.



nota

También puede abreviar el argumento del tipo de recurso `configmap` como `cm` en la interfaz de línea de comandos `oc`. Por ejemplo:

```
[user@host ~]$ oc create cm myconf --from-literal key1=value1
[user@host ~]$ oc get cm myconf
```

Para crear un nuevo secreto que almacene el contenido de un archivo o directorio que contiene un conjunto de archivos:

```
[user@host ~]$ oc create secret generic secret_name \
> --from-file /home/demo/mysecret.txt
```

Cuando crea un secreto de un archivo o directorio, los nombres de las claves se definen de la misma manera que los mapas de configuración.

Para obtener más detalles, incluido el almacenamiento de certificados TLS y claves en secretos, ejecute los comandos `oc create secret --help` y `oc secret`.

Uso de la consola web de OpenShift

También puede usar la consola web de OpenShift para crear secretos y mapas de configuración. Para crear y administrar secretos desde la consola web, inicie sesión en la consola web de OpenShift y diríjase a la página **Workloads** (Cargas de trabajo) → **Secrets** (Secretos).

| Name | Namespace | Type | Size | Created | Actions |
|-------------------------|--------------|-------------------------------------|------|-----------------|---------|
| builder-dockercfg-tbf7z | your-project | kubernetes.io/dockercfg | 1 | Aug 3, 12:05 am | ... |
| builder-token-cs2r4 | your-project | kubernetes.io/service-account-token | 4 | Aug 3, 12:05 am | ... |
| builder-token-hxj2h | your-project | kubernetes.io/service-account-token | 4 | Aug 3, 12:05 am | ... |
| default-dockercfg-97qrn | your-project | kubernetes.io/dockercfg | 1 | Aug 3, 12:05 am | ... |
| default-token-c892t | your-project | kubernetes.io/service-account-token | 4 | Aug 3, 12:05 am | ... |

Figura 2.2: Gestión de secretos desde la consola web

Para crear y administrar mapas de configuración desde la consola web, diríjase a la página **Workloads** (Cargas de trabajo) → **Config Maps** (Mapas de configuración).

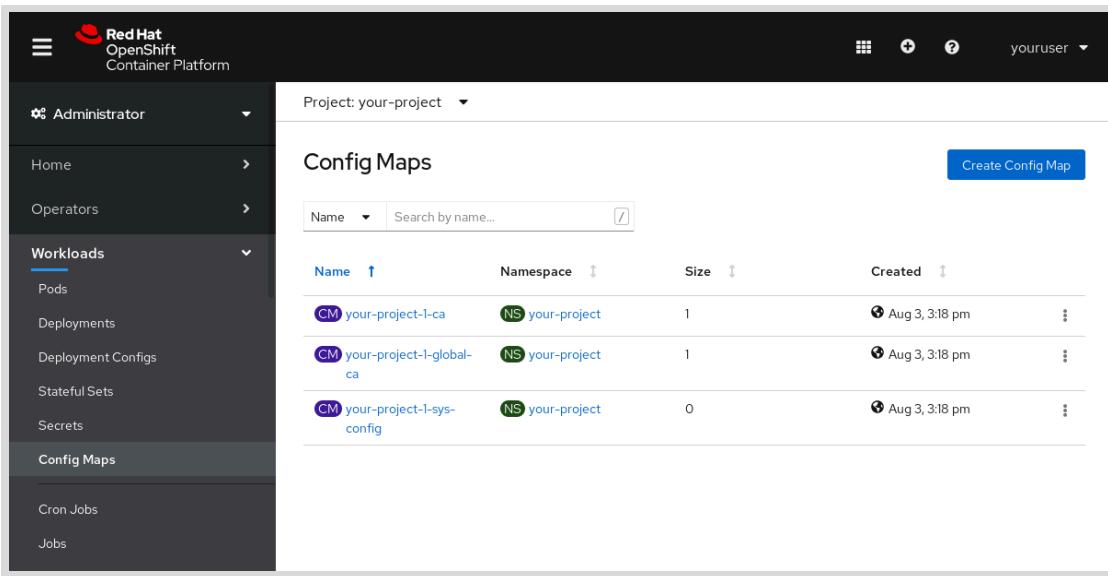


Figura 2.3: Gestión de mapas de configuración desde la consola web

Puede editar el valor asignado a cada clave de un mapa de configuración, así como el valor codificado asignado a cada clave de un secreto, mediante el editor YAML proporcionado por la consola web. Sin embargo, en el caso de un secreto, debe codificar los datos en formato base64 antes de insertarlos en la definición del recurso de secretos.

Definiciones de recursos de secretos y mapas de configuración

Dado que los mapas de configuración y los secretos son recursos comunes de OpenShift, puede usar el comando `oc create` o la consola web para importar estos archivos de definición de recursos en formato YAML o JSON.

A continuación se muestra una definición del recurso de mapa de configuración en formato YAML:

```
apiVersion: v1
data:
  key1: ① value1 ②
  key2: ③ value2 ④
kind: ConfigMap ⑤
metadata:
  name: myconf ⑥
```

- ① El nombre de la primera clave. De manera predeterminada, la variable de un entorno o un archivo con el mismo nombre que la clave se inserta en el contenedor según si el recurso de mapa de configuración se inserta como una variable de entorno o un archivo.
- ② El valor almacenado para la primera clave del mapa de configuración.
- ③ El nombre de la segunda clave.
- ④ El valor almacenado para la segunda clave del mapa de configuración.
- ⑤ El tipo de recurso de OpenShift; en este caso, un mapa de configuración.
- ⑥ Un nombre único para este mapa de configuración dentro de un proyecto.

A continuación se muestra un recurso secreto de muestra en formato YAML:

```

apiVersion: v1
data:
  username: ❶ cm9vdAo= ❷
  password: ❸ c2VjcmV0Cg== ❹
kind: Secret ❺
metadata:
  name: mysecret ❻
  type: Opaque

```

- ❶ El nombre de la primera clave. Esto proporciona el nombre predeterminado para una variable de entorno o un archivo de un pod, de la misma manera que los nombres de clave de un mapa de configuración.
- ❷ El valor almacenado para la primera clave, en formato codificado base64.
- ❸ El nombre de la segunda clave.
- ❹ El valor almacenado para la segunda clave, en formato codificado base64.
- ❺ El tipo de recurso de OpenShift; en este caso, un secreto.
- ❻ Un nombre único para este recurso secreto dentro de un proyecto.

Sintaxis alternativa para las definiciones del recurso secreto

Una plantilla no puede definir secretos mediante la sintaxis estándar debido a que todos los valores de las claves están codificados. OpenShift proporciona una sintaxis alternativa para este escenario, donde el atributo `stringData` reemplaza al atributo `data`, y los valores de la clave no están cifrados.

Con la sintaxis alternativa, el ejemplo anterior se convierte en lo siguiente:

```

apiVersion: v1
stringData:
  username: user1
  password: pass1
kind: Secret
metadata:
  name: mysecret
  type: Opaque

```

La sintaxis alternativa nunca se guarda en la base de datos etcd maestra de OpenShift. OpenShift convierte los recursos secretos definidos con la sintaxis alternativa en la representación estándar para el almacenamiento. Si ejecuta `oc get` con un secreto que se creó con la sintaxis alternativa, obtiene un recurso con la sintaxis estándar.

Comandos para manipular mapas de configuración

Para ver los detalles de un mapa de configuración en formato JSON o para exportar la definición de un recurso de mapa de configuración a un archivo JSON para la creación sin conexión:

```
[user@host ~]$ oc get configmap/myconf -o json
```

Para eliminar un mapa de configuración:

```
[user@host ~]$ oc delete configmap/myconf
```

Para editar un mapa de configuración, use el comando `oc edit`. De manera predeterminada, este comando abre un búfer similar a Vim, con la definición del recurso de mapa de configuración en formato YAML:

```
[user@host ~]$ oc edit configmap/myconf
```

Para editar un recurso de mapa de configuración, use el comando `oc patch`. Este enfoque no es interactivo y es útil cuando necesita escribir los cambios en un recurso:

```
[user@host ~]$ oc patch configmap/myconf --patch '{"data":{"key1":"newValue1"}}'
```

Comandos para manipular secretos

Los comandos para manipular recursos de secretos son similares a los que se usan para los recursos de mapa de configuración.

Para ver o exportar los detalles de un secreto:

```
[user@host ~]$ oc get secret/mysecret -o json
```

Para eliminar un secreto:

```
[user@host ~]$ oc delete secret/mysecret
```

Para editar un secreto, primero codifique sus datos en formato base64; por ejemplo:

```
[user@host ~]$ echo 'newpassword' | base64  
bmV3cGFzc3dvcmQK
```

Use el valor codificado para actualizar el recurso de secreto mediante el comando `oc edit`:

```
[user@host ~]$ oc edit secret/mysecret
```

También puede editar un recurso secreto mediante el comando `oc patch`:

```
[user@host ~]$ oc patch secret/mysecret --patch \  
> '{"data":{"password":"bmV3cGFzc3dvcmQK"}}'
```

Los secretos y mapas de configuración también pueden cambiarse y eliminarse con la consola web de OpenShift.

Inserción de datos de secretos y mapas de configuración en aplicaciones

Los secretos y mapas de configuración se pueden montar como volúmenes de datos o exponer como variables de entorno, dentro de un contenedor de aplicaciones.

Para insertar todos los valores almacenados de un mapa de configuración en variables de entorno para pods creados a partir de una configuración de implementación, use el comando `oc set env`:

```
[user@host ~]$ oc set env dc/mydcname \
> --from configmap/myconf
```

Para montar todas las claves de un mapa de configuración como archivos de un volumen dentro de pods creados a partir de una configuración de implementación, use el comando `oc set volume`:

```
[user@host ~]$ oc set volume dc/mydcname --add \
> -t configmap -m /path/to/mount/volume \
> --name myvol --configmap-name myconf
```

Para insertar datos dentro de un secreto en pods creados a partir de una configuración de implementación, use el comando `oc set env`:

```
[user@host ~]$ oc set env dc/mydcname \
> --from secret/mysecret
```

Para insertar datos de un recurso de secreto como volumen dentro de pods creados a partir de una configuración de implementación, use el comando `oc set volume`:

```
[user@host ~]$ oc set volume dc/mydcname --add \
> -t secret -m /path/to/mount/volume \
> --name myvol --secret-name mysecret
```

Cambios de secretos y mapas de configuración

Cada vez que cambia una configuración de implementación, con comandos como `oc set env` y `oc set volume`, se desencadena una nueva implementación de manera predeterminada.

Si realiza cambios en la misma configuración de implementación, le recomendamos deshabilitar los desencadenadores de cambio de configuración con el comando `oc set triggers`:

```
[user@host ~]$ oc set triggers dc/mydcname --from-config --remove
```

Después de realizar todos los cambios en los secretos o mapas de configuración, use el comando `oc rollout latest` para desencadenar una nueva implementación de la aplicación:

```
[user@host ~]$ oc rollout latest mydcname
```

Use el comando `oc set triggers` para volver a habilitar los desencadenadores:

```
[user@host ~]$ oc set triggers dc/mydcname --from-config
```

Opciones de configuración de aplicaciones

Use mapas de configuración para almacenar datos de configuración en texto sin formato y si la información no es confidencial. Use secretos si la información que almacena es confidencial.

Si su aplicación solo tiene algunas variables de configuración simples que pueden leerse desde variables de entorno o enviarse en la línea de comandos, use variables de entorno para insertar

datos de secretos y mapas de configuración. El enfoque preferido son las variables de entorno por sobre el montaje de volúmenes dentro del contenedor.

En cambio, si su aplicación tiene una gran cantidad de variables de configuración, o si está migrando una aplicación heredada que usa muchos archivos de configuración, use el enfoque de montaje de volumen en vez de crear una variable de entorno para cada una de las variables de configuración. Por ejemplo, si la aplicación espera uno o más archivos de configuración de una ubicación específica del sistema de archivos, debe crear secretos o mapas de configuración a partir de los archivos de configuración y montarlos dentro del sistema de archivos efímeros del contenedor en la ubicación que se espera la aplicación.

Para lograr ese objetivo, con secretos que apuntan al archivo `/home/student/configuration.properties`, use el siguiente comando:

```
[user@host ~]$ oc create secret generic security \
> --from-file /home/student/configuration.properties
```

Para insertar el secreto en la aplicación, configure un volumen que haga referencia a los secretos creados en el comando anterior. El volumen debe apuntar a un directorio real dentro de la aplicación donde se almacena el archivo de secretos.

En el ejemplo siguiente, el archivo `configuration.properties` se almacena en el directorio `/opt/app-root/secure`. Para enlazar el archivo a la aplicación, ajuste la configuración de implementación desde la aplicación (`dc/application`):

```
[user@host ~]$ oc set volume dc/application --add \
> -t secret -m /opt/app-root/secure \
> --name myappsec-vol --secret-name security
```

Para crear un mapa de configuración, use el siguiente comando:

```
[user@host ~]$ oc create configmap properties \
> --from-file /home/student/configuration.properties
```

Para enlazar la aplicación al mapa de configuración, actualice la configuración de implementación de esa aplicación para usar el mapa de configuración:

```
[user@host ~]$ oc set env dc/application \
> --from configmap/properties
```



Referencias

Puede obtener más información sobre los secretos en el capítulo *Provisión de datos confidenciales a pods* de la guía *Nodos para Red Hat OpenShift Container Platform 4.5* en

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/nodes/working-with-pods#nodes-pods-secrets

► Ejercicio Guiado

Inserción de datos de configuración en una aplicación

En este ejercicio, usará secretos y mapas de configuración para externalizar la configuración de una aplicación contenedora.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implementar una aplicación simple basada en Node.js que imprima detalles de configuración de archivos y variables de entorno.
- Insertar datos de configuración en un contenedor mediante secretos y mapas de configuración.
- Cambiar los datos del mapa de configuración y verificar que la aplicación reciba los valores cambiados.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I para Node.js 12.
- La aplicación de muestra del repositorio Git (`app-config`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos del ejercicio y descargar los archivos de trabajo y soluciones:

```
[student@workstation ~]$ lab app-config start
```

► 1. Revise el código fuente de la aplicación.

- 1.1. Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación `master` del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 1.2. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b app-config
Switched to a new branch 'app-config'
[student@workstation D0288-apps]$ git push -u origin app-config
...output omitted...
* [new branch]      app-config -> app-config
Branch app-config set up to track remote branch app-config from origin.
```

- 1.3. Inspeccione el archivo /home/student/D0288-apps/app-config/app.js.

La aplicación lee el valor de la variable de entorno APP_MSG e imprime el contenido del archivo /opt/app-root/secure/myapp.sec:

```
// read in the APP_MSG env var
var msg = process.env.APP_MSG;
...output omitted...
// Read in the secret file
fs.readFile('/opt/app-root/secure/myapp.sec', 'utf8', function (secerr, secdata) {
...output omitted...
```

► 2. Compile e implemente la aplicación.

- 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 D0288-apps]$ source /usr/local/etc/ocp4.config
```

- 2.2. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 2.3. Cree un nuevo proyecto para la aplicación. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-app-config
```

- 2.4. Cree una nueva aplicación denominada myapp a partir de las fuentes en Git. Use la bifurcación que creó en el paso anterior.

Puede copiar o ejecutar el comando del script oc-new-app.sh en la carpeta /home/student/D0288/labs/app-config:

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config --name myapp \
> --build-env npm_config_registry=
> http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs \
> nodejs:12-https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#app-config \
> --context-dir app-config
...output omitted...
```

```
--> Creating resources ...output omitted...
imagestream.image.openshift.io "myapp" created
buildconfig.build.openshift.io "myapp" created
deploymentconfig.apps.openshift.io "myapp" created
service "myapp" created
--> Success
...output omitted...
```

Observe que no hay espacio antes ni después del signo igual (=) después de `npm_config_registry`.

- 2.5. Vea los registros de compilación. Espere hasta que termine la compilación y la imagen del contenedor de aplicaciones se envíe al registro interno de OpenShift:

```
[student@workstation D0288-apps]$ oc logs -f bc/myapp
Cloning "https://github.com/youruser/D0288-apps#app-config" ...
...output omitted...
---> Installing application source ...
---> Building your Node application from source
...output omitted...
Pushing image image-registry.openshift-image-registry.svc:5000/youruser-app-
config/myapp:latest ...
...output omitted...
Push successful
```

► 3. Pruebe la aplicación.

- 3.1. Espere a que se implemente la aplicación. Consulte el estado del pod de la aplicación. El pod de la aplicación debe estar en el estado **Running** (En ejecución):

```
[student@workstation D0288-apps]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
myapp-1-2w1nc  1/1     Running   0          81s
myapp-1-deploy 0/1     Completed  0          89s
myapp-1-build  0/1     Completed  0          3m
```

- 3.2. Use una ruta para exponer la aplicación a acceso externo:

```
[student@workstation D0288-apps]$ oc expose svc myapp
route.route.openshift.io/myapp exposed
```

- 3.3. Identifique la URL de enrutamiento donde se expone la API de la aplicación:

```
[student@workstation D0288-apps]$ oc get route
NAME      HOST/PORT           ...
myapp    myapp-youruser-app-config.apps.cluster.domain.example.com ...
```

- 3.4. Invoque la URL de enrutamiento identificada en el paso anterior con el comando `curl`:

```
[student@workstation D0288-apps]$ curl \
> http://myapp-${RHT_OCP4_DEV_USER}-app-config.${RHT_OCP4_WILDCARD_DOMAIN}
Value in the APP_MSG env var is => undefined
Error: ENOENT: no such file or directory, open '/opt/app-root/secure/myapp.sec'
```

El valor undefined para la variable de entorno, y el error ENOENT: no such file or directory (ENOENT: no existe dicho archivo o directorio) aparecen porque no hay una variable de entorno o un archivo en el contenedor.

- 4. Cree los recursos de secretos y mapa de configuración.
- 4.1. Cree un recurso de mapa de configuración para guardar las variables de configuración que almacenan datos de texto sin formato.
Cree un nuevo recurso de mapa de configuración denominado myappconf. Almacene una clave denominada APP_MSG con el valor TestMessage en este mapa de configuración:

```
[student@workstation D0288-apps]$ oc create configmap myappconf \
> --from-literal APP_MSG="Test Message"
configmap/myappconf created
```

- 4.2. Verifique que el mapa de configuración contenga los datos de configuración:

```
[student@workstation D0288-apps]$ oc describe cm/myappconf
Name: myappconf
...output omitted...
Data
=====
APP_MSG:
-----
Test Message
...output omitted...
```

- 4.3. Revise el contenido del archivo /home/student/D0288-apps/app-config/myapp.sec:

```
username=user1
password=pass1
salt=xyz123
```

- 4.4. Cree un nuevo secreto para almacenar el contenido del archivo myapp.sec.

```
[student@workstation D0288-apps]$ oc create secret generic myappfilesec \
> --from-file /home/student/D0288-apps/app-config/myapp.sec
secret/myappfilesec created
```

- 4.5. Verifique el contenido del secreto. Observe que el contenido está almacenado en el formato codificado base64:

```
[student@workstation D0288-apps]$ oc get secret/myappfilesec -o json
{
  "apiVersion": "v1",
  "data": {
    "myapp.sec": "dXNlcj5hbWU9dXNlcjEKcGFzc3dvcmQ9cGFzcxEKc2..."
  },
  "kind": "Secret",
  "metadata": {
    ...output omitted...
    "name": "myappfilesec",
    ...output omitted...
  },
  "type": "Opaque"
}
```

► 5. Inserte el mapa de configuración y el secreto en el contenedor de aplicaciones.

- 5.1. Use el comando `oc set env` para agregar el mapa de configuración a la configuración de implementación:

```
[student@workstation ~]$ oc set env dc/myapp \
> --from configmap/myappconf
deploymentconfig.apps.openshift.io/myapp updated
```

- 5.2. Use el comando `oc set volume` para agregar el secreto a la configuración de implementación:

Puede copiar o ejecutar el comando del script `inject-secret-file.sh` en la carpeta `/home/student/D0288/labs/app-config`:

```
[student@workstation D0288-apps]$ oc set volume dc/myapp --add \
> -t secret -m /opt/app-root/secure \
> --name myappsec-vol --secret-name myappfilesec
deploymentconfig.apps.openshift.io/myapp volume updated
```

► 6. Verifique que la aplicación se vuelva a implementar y use los datos del secreto y del mapa de configuración.

- 6.1. Verifique que la aplicación se vuelva a implementar debido a los cambios realizados en la configuración de implementación en los pasos anteriores:

```
[student@workstation D0288-apps]$ oc status
In project youruser-app-config on server ...output omitted...

http://myapp-youruser-app-config.apps.cluster.domain.example.com to pod port 8080-
tcp (svc/myapp)
dc/myapp deploys istag/myapp:latest <
  bc/myapp source builds https://github.com/youruser/D0288-apps#app-config on
  openshift/nodejs:12
  deployment #3 deployed 24 seconds ago - 1 pod
  deployment #2 deployed 4 minutes ago
  deployment #1 deployed 18 minutes ago
```

**nota**

Debe ver que la aplicación se volvió a implementar dos veces, debido a los dos comandos `oc set env` que cambian la configuración de implementación.

También puede ignorar los mensajes de error similares a los siguientes:

```
deployment #2 failed 59 seconds ago: newer deployment was found running
```

- 6.2. Espere hasta que el pod de aplicación esté listo y tenga el estado Running (En ejecución). Obtenga el nombre del pod de aplicación con el comando `oc get pods`.

```
[student@workstation D0288-apps]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
myapp-1-build  0/1    Completed  0          13m
myapp-1-deploy 0/1    Completed  0          12m
myapp-3-deploy 0/1    Completed  0          3m
myapp-3-wzdbh  1/1    Running   0          2m
```

- 6.3. Use el comando `oc rsh` para inspeccionar las variables de entorno en el contenedor:

```
[student@workstation D0288-apps]$ oc rsh myapp-3-wzdbh env | grep APP_MSG
APP_MSG=Test Message
```

- 6.4. Verifique que el mapa de configuración y el secreto se hayan insertado en el contenedor. Vuelva a probar la aplicación con la URL de enrutamiento:

```
[student@workstation D0288-apps]$ curl \
> http://myapp-${RHT_OCP4_DEV_USER}-app-config.${RHT_OCP4_WILDCARD_DOMAIN}
Value in the APP_MSG env var is => Test Message
The secret is => username=user1
password=pass1
salt=xyz123
```

OpenShift inserta el mapa de configuración como una variable de entorno y monta el secreto como un archivo en el contenedor. La aplicación lee la variable de entorno y el archivo, y muestra sus datos.

- 7. Cambie la información almacenada en el mapa de configuración y vuelva a probar la aplicación.

- 7.1. Use el comando `oc edit configmap` para cambiar el valor de la clave APP_MSG:

```
[student@workstation D0288-apps]$ oc edit cm/myappconf
```

El comando anterior abre un búfer similar a Vim con los atributos del mapa de configuración en formato YAML. Edite el valor asociado con la clave APP_MSG en la sección de datos y cambie el valor de la siguiente manera:

```
...output omitted...
apiVersion: v1
data:
  APP_MSG: Changed Test Message
kind: ConfigMap
...output omitted...
```

Guarde y cierre el archivo.

- 7.2. Verifique que el valor en la clave APP_MSG se haya actualizado:

```
[student@workstation D0288-apps]$ oc describe cm/myappconf
Name: myappconf
...output omitted...
Data
=====
APP_MSG:
-----
Changed Test Message
...output omitted...
```

- 7.3. Use el comando `oc rollout latest` para desencadenar una nueva implementación. Esto garantiza que la aplicación reciba los valores cambiados en el mapa de configuración:

```
[student@workstation D0288-apps]$ oc rollout latest dc/myapp
deploymentconfig.apps.openshift.io/myapp rolled out
```

- 7.4. Espere a que el pod de aplicación vuelva a implementarse y tenga el estado Running (En ejecución):

```
[student@workstation D0288-apps]$ oc get pods
NAME        READY     STATUS    RESTARTS   AGE
...output omitted...
myapp-4-2feqr  1/1      Running     0          6s
```

- 7.5. Pruebe la aplicación y verifique que se muestren los valores cambiados en el mapa de configuración:

```
[student@workstation D0288-apps]$ curl \
> http://myapp-${RHT_OCP4_DEV_USER}-app-config.${RHT_OCP4_WILDCARD_DOMAIN}
Value in the APP_MSG env var is => Changed Test Message
The secret is => username=user1
password=pass1
salt=xyz123
```

- 8. Realice la limpieza. Elimine el proyecto `youruser-app-config` de OpenShift.

```
[student@workstation D0288-apps]$ oc delete project \
> ${RHT_OCP4_DEV_USER}-app-config
project.project.openshift.io "youruser-app-config" deleted
```

Finalizar

En `workstation`, ejecute el comando `lab app-config finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation DO288-apps]$ lab app-config finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Diseño de aplicaciones contenerizadas para OpenShift

Lista de verificación de rendimiento

En este trabajo de laboratorio, corregirá el Dockerfile para que una aplicación basada en el marco (framework) Thorntail se ejecute en un clúster OpenShift. También usará un mapa de configuración para configurar la aplicación.



nota

El comando `grade` al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se declaran en la especificación del trabajo de laboratorio.

Resultados

Deberá ser capaz de corregir el Dockerfile para que una aplicación basada en el marco (framework) Thorntail se ejecute como usuario aleatorio y, luego, implementar la aplicación en un clúster OpenShift. También deberá ser capaz de usar un mapa de configuración para almacenar una cadena de texto simple usada para configurar la aplicación.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- El fat JAR ejecutable de la aplicación.
- El repositorio Git de la aplicación.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos. El comando también descarga archivos auxiliares y de solución para el trabajo de laboratorio:

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

Requisitos

La aplicación está programada en Java, con el marco (framework) Thorntail. Se ofrece el archivo JAR ejecutable que fue compilado previamente (fat JAR), el cual contiene la aplicación, y el tiempo de ejecución Thorntail. La aplicación ofrece una API REST simple que responde a solicitudes según una configuración que se inyecta en el contenedor como variable de entorno. Compile e implemente la aplicación en un clúster OpenShift, según los siguientes requisitos:

- El nombre de la aplicación para OpenShift es `elvis`. La configuración para la aplicación debe estar almacenada en un mapa de configuración denominado `appconfig`.
- Implemente la aplicación en un proyecto denominado `youruser-design-container`.

- La API REST para la aplicación debe estar accesible en la siguiente URL:
`elvis-youruser-design-container.apps.cluster.domain.example.com/api/hello.`
- Este es el repositorio Git (carpeta) que contiene las fuentes del directorio de aplicaciones:
`https://github.com/youruser/D0288-apps/hello-java.`
- El archivo JAR de la aplicación compilada previamente está disponible en:
`https://github.com/RedHatTraining/D0288-apps/releases/download/OCP-4.1-1/hello-java.jar`

Pasos

1. Navegue a su clon local del repositorio Git D0288-apps y cree una nueva bifurcación denominada `design-container` de la bifurcación maestra. Revise brevemente el Dockerfile de la aplicación en el directorio `/home/student/D0288-apps/hello-java/`.
2. Implemente la aplicación en la carpeta `hello-java` del repositorio Git D0288-apps, con la bifurcación `design-container` de la aplicación. Implemente la aplicación en el proyecto `youruser-design-container` en OpenShift sin realizar ningún cambio.
No olvide obtener las variables del archivo `/usr/local/etc/ocp4.config` antes de iniciar sesión en el clúster OpenShift.
3. Siga los registros de compilación y verifique que la imagen de contenedor haya sido compilada y enviada al registro interno de OpenShift. Vea el estado de implementación del pod de la aplicación. El pod se encontrará en un estado de `CrashLoopBackOff` o `Error`. Vea los registros de la aplicación para ver por qué la aplicación no se inicia correctamente.
4. Edite el Dockerfile de la aplicación para garantizar su implementación exitosa en un clúster OpenShift. El contenedor debe ejecutarse con un identificador de usuario aleatorio en lugar del usuario `wildfly` configurado actualmente.
5. Confirme los cambios que realizó en el Dockerfile y envíe los cambios al repositorio Git del aula.
6. Inicie una nueva compilación de la aplicación. Siga el registro de compilación de la nueva compilación. Verifique que el pod de la aplicación se inicie correctamente.
7. Exponga el servicio al acceso externo y pruebe la aplicación. Acceda a la API de la aplicación mediante la ruta de contexto `/api/hello`.
8. Cree un nuevo mapa de configuración denominado `appconfig`. Almacene una clave denominada `APP_MSG` con el valor `Elvis lives` en este mapa de configuración. Agregue esa clave como una variable de entorno a la configuración de implementación de la aplicación.
9. Verifique que se desencadene una nueva implementación y espere a que un nuevo pod de aplicación esté listo y en ejecución. Verifique que la clave `APP_MSG` se inserte en el contenedor como variable de entorno.
10. Pruebe la aplicación invocando la URL de la API REST (`http://elvis-youruser-design-container.apps.cluster.domain.example.com/api/hello`) y verifique que el valor de la clave `APP_MSG` aparezca en la respuesta.

Evaluación

Inicie sesión con el usuario `student` en la máquina `workstation` y use el comando `lab` para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

En `workstation`, ejecute el comando `lab design-container finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

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

Esto concluye el trabajo de laboratorio.

► Solución

Diseño de aplicaciones contenerizadas para OpenShift

Listado de verificación de rendimiento

En este trabajo de laboratorio, corregirá el Dockerfile para que una aplicación basada en el marco (framework) Thorntail se ejecute en un clúster OpenShift. También usará un mapa de configuración para configurar la aplicación.



nota

El comando `grade` al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se declaran en la especificación del trabajo de laboratorio.

Resultados

Deberá ser capaz de corregir el Dockerfile para que una aplicación basada en el marco (framework) Thorntail se ejecute como usuario aleatorio y, luego, implementar la aplicación en un clúster OpenShift. También deberá ser capaz de usar un mapa de configuración para almacenar una cadena de texto simple usada para configurar la aplicación.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- El fat JAR ejecutable de la aplicación.
- El repositorio Git de la aplicación.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos. El comando también descarga archivos auxiliares y de solución para el trabajo de laboratorio:

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

Requisitos

La aplicación está programada en Java, con el marco (framework) Thorntail. Se ofrece el archivo JAR ejecutable que fue compilado previamente (fat JAR), el cual contiene la aplicación, y el tiempo de ejecución Thorntail. La aplicación ofrece una API REST simple que responde a solicitudes según una configuración que se inyecta en el contenedor como variable de entorno. Compile e implemente la aplicación en un clúster OpenShift, según los siguientes requisitos:

- El nombre de la aplicación para OpenShift es `elvis`. La configuración para la aplicación debe estar almacenada en un mapa de configuración denominado `appconfig`.
- Implemente la aplicación en un proyecto denominado `youruser-design-container`.

- La API REST para la aplicación debe estar accesible en la siguiente URL:
`elvis-youruser-design-container.apps.cluster.domain.example.com/api/hello.`
- Este es el repositorio Git (carpeta) que contiene las fuentes del directorio de aplicaciones:
`https://github.com/youruser/D0288-apps/hello-java.`
- El archivo JAR de la aplicación compilada previamente está disponible en:
`https://github.com/RedHatTraining/D0288-apps/releases/download/OCP-4.1-1/hello-java.jar`

Pasos

1. Navegue a su clon local del repositorio Git D0288-apps y cree una nueva bifurcación denominada `design-container` de la bifurcación maestra. Revise brevemente el Dockerfile de la aplicación en el directorio `/home/student/D0288-apps/hello-java/`.

- 1.1. Consulte la bifurcación maestra del repositorio Git.

```
[student@workstation ~]$ cd D0288-apps  
[student@workstation D0288-apps]$ git checkout master  
...output omitted...
```

- 1.2. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b design-container  
Switched to a new branch 'design-container'  
[student@workstation D0288-apps]$ git push -u origin design-container  
...output omitted...  
* [new branch]      design-container -> design-container  
Branch design-container set up to track remote branch design-container from  
origin.
```

- 1.3. Inspeccione el archivo `/home/student/D0288-apps/hello-java/Dockerfile`. Por el momento, no lo modifique.
2. Implemente la aplicación en la carpeta `hello-java` del repositorio Git D0288-apps, con la bifurcación `design-container` de la aplicación. Implemente la aplicación en el proyecto `youruser-design-container` en OpenShift sin realizar ningún cambio.

No olvide obtener las variables del archivo `/usr/local/etc/ocp4.config` antes de iniciar sesión en el clúster OpenShift.

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

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

```
[student@workstation D0288-apps]$ source /usr/local/etc/ocp4.config
```

- 2.2. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 2.3. Cree un nuevo proyecto para la aplicación. Coloque como prefijo del nombre del proyecto su nombre de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-design-container
```

- 2.4. Cree una nueva aplicación denominada elvis a partir del Dockerfile almacenado en el repositorio Git.

Copie o ejecute el comando del script `oc-new-app.sh` en el directorio `/home/student/D0288/labs/design-container`.

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config --name elvis \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#design-container \
> --context-dir hello-java
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "ubi" created
imagestream.image.openshift.io "elvis" created
buildconfig.build.openshift.io "elvis" created
deploymentconfig.apps.openshift.io "elvis" created
service "elvis" created
--> Success
...output omitted...
```

3. Siga los registros de compilación y verifique que la imagen de contenedor haya sido compilada y enviada al registro interno de OpenShift. Vea el estado de implementación del pod de la aplicación. El pod se encontrará en un estado de CrashLoopBackOff o Error. Vea los registros de la aplicación para ver por qué la aplicación no se inicia correctamente.

- 3.1. Siga los registros de compilación:

```
[student@workstation D0288-apps]$ oc logs -f bc/elvis
...output omitted...
STEP 1: FROM registry.access.redhat.com/ubi8/ubi@sha256...
...output omitted...
Pushing image image-registry.openshift-image-registry.svc:5000/youruser-design-
container/elvis:latest ...
...output omitted...
Push successful
```

- 3.2. Espere hasta que el pod de la aplicación se haya implementado. La aplicación no alcanza un estado Ready. Después de algún tiempo, permanecerá en un estado CrashLoopBackOff o Error.

```
[student@workstation D0288-apps]$ oc get pods
NAME        READY   STATUS            RESTARTS   AGE
...output omitted...
elvis-1-tgv5s   0/1     CrashLoopBackOff   1          13s
```

- 3.3. Vea los registros para el pod de la aplicación e investigue por qué el pod de la aplicación no logró iniciarse.

```
[student@workstation D0288-apps]$ oc logs elvis-1-tgv5s
/bin/sh: /opt/app-root/bin/run-app.sh: Permission denied
```

La aplicación no se inicia debido a un error de "Permiso denegado" en el sistema de archivos, porque OpenShift no ejecuta el pod con el usuario especificado en el Dockerfile.

4. Edite el Dockerfile de la aplicación para garantizar su implementación exitosa en un clúster OpenShift. El contenedor debe ejecutarse con un identificador de usuario aleatorio en lugar del usuario `wildfly` configurado actualmente.

- 4.1. Edite el Dockerfile en `/home/student/D0288-apps/hello-java/Dockerfile` con un editor de texto. Realice los cambios descritos en los siguientes pasos. También puede copiar las instrucciones y los comandos del archivo de solución provisto en `/home/student/D0288/solutions/design-container/Dockerfile`.

Elimine el comando `useradd` de la primera instrucción RUN (línea 12).

```
useradd wildfly && \
```

- 4.2. Ubique los comandos `chown` y `chmod` en las líneas 19 y 20:

```
RUN chown -R wildfly:wildfly /opt/app-root && \
    chmod -R 700 /opt/app-root
```

Reemplácelos con lo siguiente:

```
RUN chgrp -R 0 /opt/app-root && \
    chmod -R g=u /opt/app-root
```

- 4.3. Reemplace el usuario `wildfly` en la instrucción `USER` de la línea 24 con el ID de usuario genérico 1001 para evitar heredar el usuario de la imagen RHEL principal. OpenShift ignora este ID de usuario genérico y sigue las recomendaciones y convenciones de Red Hat para la compilación de imágenes:

```
USER 1001
```

5. Confirme los cambios que realizó en el Dockerfile y envíe los cambios al repositorio Git del aula.

```
[student@workstation DO288-apps]$ cd hello-java
[student@workstation hello-java]$ git commit -a -m \
> "Fixed Dockerfile to run with random user id on OpenShift"
[student@workstation hello-java]$ git push
[student@workstation hello-java]$ cd ~
[student@workstation ~]$
```

6. Inicie una nueva compilación de la aplicación. Siga el registro de compilación de la nueva compilación. Verifique que el pod de la aplicación se inicie correctamente.

- 6.1. Inicie una nueva compilación para la aplicación:

```
[student@workstation ~]$ oc start-build elvis
build.build.openshift.io/elvis-2 started
```

- 6.2. Siga el registro de compilación y verifique que se haya creado y se haya enviado una nueva imagen de contenedor al registro interno OpenShift:

```
[student@workstation ~]$ oc logs -f bc/elvis
...output omitted...
Pushing image image-registry.openshift-image-registry.svc:5000/youruser-design-
container/elvis:latest ...
...output omitted...
Push successful
```

- 6.3. Espere a que el pod de la aplicación esté listo y en ejecución.

| NAME | READY | STATUS | RESTARTS | AGE |
|---------------|-------|---------|----------|-----|
| elvis-2-9gvqr | 1/1 | Running | 0 | 9s |

- 6.4. Vea los registros para el pod de la aplicación y verifique que no haya errores durante el inicio:

```
[student@workstation ~]$ oc logs elvis-2-9gvqr
Starting hello-java app...
JVM options => -Xmx512m
...output omitted...
2017-10-12 13:49:20,647 INFO [org.jboss.as] (MSC service thread 1-3) WFLYSRV0049:
Thorntail 2.4.0.Final (WildFly Core 7.0.0.Final) starting
...output omitted...
2017-10-12 13:49:23,237 INFO [org.wildfly.swarm] (main) THORN99999: Thorntail is
Ready
```

7. Exponga el servicio al acceso externo y pruebe la aplicación. Acceda a la API de la aplicación mediante la ruta de contexto /api/hello.

- 7.1. Exponga la aplicación para acceso externo.

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

- 7.2. Identifique el nombre de host donde se expone la API de la aplicación:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT
elvis     elvis-youruser-design-container.apps.cluster.domain.example.com
```

- 7.3. Pruebe la aplicación invocando la URL de la API con el nombre de host identificado en el paso anterior (<http://elvis-youruser-design-container.apps.cluster.domain.example.com/api/hello>) y verifique que el nombre del pod de la aplicación aparezca en la respuesta:

```
[student@workstation ~]$ curl \
> http://elvis-${RHT_OCP4_DEV_USER}-design-container.${RHT_OCP4_WILDCARD_DOMAIN}\
> /api/hello
Hello world from host elvis-2-9gvqr
```

8. Cree un nuevo mapa de configuración denominado appconfig. Almacene una clave denominada APP_MSG con el valor `Elvis lives` en este mapa de configuración. Agregue esa clave como una variable de entorno a la configuración de implementación de la aplicación.

- 8.1. Cree el mapa de configuración:

```
[student@workstation ~]$ oc create cm appconfig \
> --from-literal APP_MSG="Elvis lives"
configmap/appconfig created
```

- 8.2. Vea los detalles del mapa de configuración:

```
[student@workstation ~]$ oc describe cm/appconfig
Name:  appconfig
...output omitted...

Data
=====
APP_MSG:
-----
Elvis lives
```

- 8.3. Use el comando `oc set env` para agregar el mapa de configuración a la configuración de implementación:

```
[student@workstation ~]$ oc set env dc/elvis --from cm/appconfig
deploymentconfig.apps.openshift.io/elvis updated
```

9. Verifique que se desencadene una nueva implementación y espere a que un nuevo pod de aplicación esté listo y en ejecución. Verifique que la clave APP_MSG se inserte en el contenedor como variable de entorno.

9.1. Verifique que se haya desencadenado la nueva implementación:

```
[student@workstation ~]$ oc status
...output omitted...
dc/elvis deploys istag/elvis:latest <-
  bc/elvis docker builds https://github.com/youruser/D0288-apps#design-container
on istag/ubi:8.0
  deployment #3 deployed 3 minutes ago - 1 pod
  deployment #2 deployed 38 minutes ago
...output omitted...
```

9.2. Espere a que se vuelva a implementar el pod de la aplicación. Verifique que el nuevo pod de aplicación esté listo y en ejecución:

```
[student@workstation ~]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
...output omitted...
elvis-3-ks1np  1/1     Running      0          3m
```

9.3. Verifique que la clave APP_MSG se inserte en el contenedor como variable de entorno:

```
[student@workstation ~]$ oc rsh elvis-3-ks1np env | grep APP_MSG
APP_MSG=Elvis lives
```

10. Pruebe la aplicación invocando la URL de la API REST (<http://elvis-youruser-design-container.apps.cluster.domain.example.com/api/hello>) y verifique que el valor de la clave APP_MSG aparezca en la respuesta.

```
[student@workstation ~]$ curl \
> http://elvis-${RHT_OCP4_DEV_USER}-design-container.${RHT_OCP4_WILDCARD_DOMAIN}\
> /api/hello
Hello world from host [elvis-3-ks1np]. Message received = Elvis lives
```

Evaluación

Inicie sesión con el usuario student en la máquina workstation y use el comando lab para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

En workstation, ejecute el comando lab design-container finish para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Entre las opciones de implementación de contenedores RHOCP se incluyen las siguientes:
 - Implementar imágenes de contenedor precompiladas directamente en un clúster OpenShift.
 - Crear un Dockerfile con una imagen base y personalizarlo para que se adapte a sus necesidades.
 - Usar una compilación de fuente a imagen (S2I) donde RHOCP combina el código fuente con una imagen de compilador.
- Cambios comunes que se deben hacer en los Dockerfiles para ejecutar un contenedor en RHOCP:
 - Permisos de grupo raíz en archivos leídos o escritos por procesos en el contenedor.
 - Los archivos ejecutables deben contar con permisos de ejecución de grupos.
 - Los procesos que se ejecutan en el contenedor no deben escuchar en puertos con privilegios (puertos inferiores a 1024).
- Use secretos para almacenar información confidencial y acceder a ella desde sus pods.
- Use los recursos de mapa de configuración para almacenar datos específicos y no confidenciales del entorno.

capítulo 3

Publicación de imágenes de contenedor empresariales

Meta

Interactuar con un registro empresarial y publicar imágenes de contenedor en él.

Objetivos

- Gestionar las imágenes de contenedor en los registros mediante las herramientas de contenedor de Linux.
- Acceder al registro interno de OpenShift mediante las herramientas de contenedor de Linux.
- Crear secuencias de imágenes para imágenes de contenedor en registros externos.

Secciones

- Administración de imágenes en un registro empresarial (y ejercicio guiado)
- Permisos de acceso al registro de OpenShift (y ejercicio guiado)
- Creación de flujos de imágenes (y ejercicio guiado)

Trabajo de laboratorio

Publicación de imágenes de contenedor empresariales

Administración de imágenes en un registro empresarial

Objetivos

Después de completar esta sección, deberá ser capaz de administrar imágenes de contenedor en registros con herramientas de contenedor de Linux.

Revisión de registros de contenedor

Un *registro de imágenes de contenedor*, *registro de contenedores* o *servidor de registro* almacena las imágenes que implementa como contenedores y proporciona mecanismos para extraer, insertar, actualizar, buscar y quitar imágenes de contenedor. Usa una API REST estándar definida por *Open Container Initiative (OCI)*, que se basa en Docker Registry HTTP API v2. Desde la perspectiva de una organización que ejecuta un clúster OpenShift, hay muchos tipos de registros de contenedor:

Registros públicos

Registros que permiten que cualquier persona consuma imágenes de contenedor directamente desde Internet sin ninguna autenticación. Docker Hub, Quay.io y el registro de Red Hat son ejemplos de registros de contenedores públicos.

Registros privados

Los registros que solo están disponibles para los consumidores seleccionados y que normalmente requieren autenticación. El registro basado en términos de Red Hat es un ejemplo de un registro de contenedores privado.

Registros externos

Registros que su organización no controla. Suelen administrarlos un proveedor de nube o un proveedor de software. Quay.io es un ejemplo de un registro de contenedor externo.

Registros empresariales

Servidores de registro que administra su organización. Por lo general, solo están disponibles para los empleados y contratistas de la organización.

Registros internos de OpenShift

Un servidor de registro administrado internamente por un clúster de OpenShift para almacenar imágenes de contenedor. Cree esas imágenes con las configuraciones de compilación de OpenShift y el proceso S2I o Dockerfile, o bien impórtelas de otros registros.

Este tipo de registros no son mutuamente excluyentes: un registro puede ser, al mismo tiempo, tanto registro público como privado. Por lo general, un registro público también es un registro externo, porque su organización puede tener acceso a él a través de Internet, sin autenticación, y su organización no lo controla. El mismo registro también podría ser un registro privado, si su organización tiene un plan con el proveedor de registros que le permite alojar imágenes privadas y su organización también tiene control sobre quién más puede tener acceso a esas imágenes privadas.

Quay.io funciona como un registro público y privado para algunos usuarios. El mismo desarrollador puede usar algunas imágenes de contenedor público de Quay.io, y también algunas imágenes de contenedor de un proveedor, lo que requiere autenticación.

Registros administrados por Red Hat

Red Hat administra un conjunto de registros de contenedores públicos y privados para suministrar diferentes tipos de imágenes de contenedor a distintos destinatarios. Las imágenes de contenedor soportadas con los SLA de nivel de producción de Red Hat o sus socios son accesibles a través de *Red Hat Container Catalog*. Se puede acceder a las imágenes de contenedor de la comunidad y no soportadas a través de Quay.io.

Red Hat Container Catalog (RHCC), en <https://access.redhat.com/containers>, es una interfaz de usuario web que le permite explorar y buscar esos registros y obtener información detallada sobre las imágenes basadas en Red Hat Enterprise Linux.

Las imágenes proporcionadas por Red Hat se benefician de la larga experiencia de Red Hat en la administración de vulnerabilidades de seguridad y defectos en Red Hat Enterprise Linux y otros productos. El equipo de seguridad de Red Hat refuerza y controla estas imágenes de alta calidad; luego, las firma para evitar su uso indebido. Red Hat también reconstruye estas imágenes cada vez que se descubren nuevas vulnerabilidades y ejecuta un proceso de control de calidad.

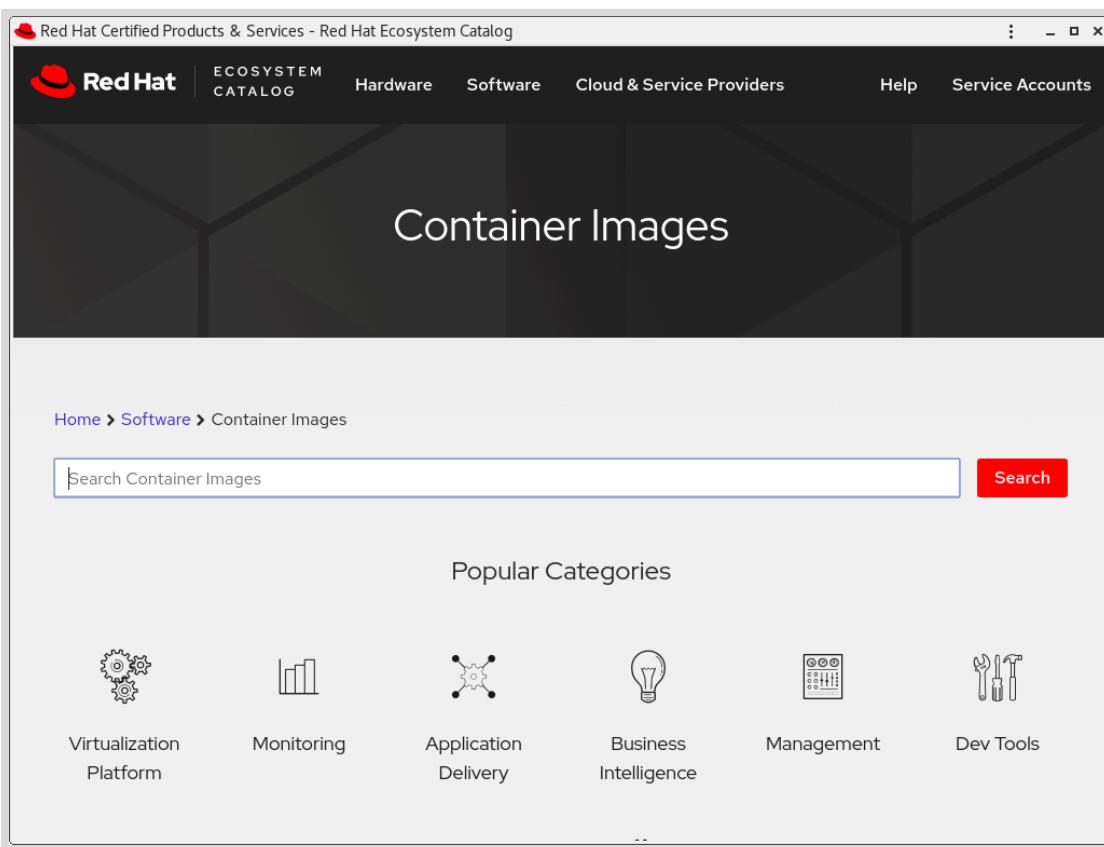


Figura 3.1: Red Hat Container Catalog

Las aplicaciones de misión crítica deben confiar en estas imágenes confiables y soportadas tanto como sea posible en lugar de usar imágenes de otros registros públicos. Esas otras imágenes pueden no probarse, mantenerse o actualizarse de manera adecuada cuando se detectan nuevos problemas de seguridad.

Red Hat Container Catalog presenta una vista unificada de los tres registros de contenedores subyacentes:

Red Hat Container Registry en registry.access.redhat.com

Es un registro público que aloja imágenes para productos de Red Hat y no requiere autenticación. Tenga en cuenta que, aunque este registro de contenedores es público, la mayoría de las imágenes de contenedor que proporciona Red Hat requieren que el usuario tenga una suscripción de producto de Red Hat activa y que cumpla con el Acuerdo de usuario final (EUA) del producto. Solo un subconjunto de las imágenes disponibles en el registro público de Red Hat se pueden redistribuir libremente. Se trata de las imágenes basadas en imágenes de base universal (UBI) de Red Hat Enterprise Linux.

Registro basado en términos de Red Hat en registry.redhat.io

Es un registro privado que aloja imágenes para productos de Red Hat y no requiere autenticación. Para extraer imágenes de ahí, debe autenticarse con sus credenciales del portal de clientes de Red Hat. Para entornos compartidos, como tuberías de OpenShift o CI/CD, puede crear una cuenta de servicio, o token de autenticación, para evitar la exposición de sus credenciales personales.

Registro de socios de Red Hat en registry.connect.redhat.com

Es un registro privado que aloja imágenes para productos de terceros de socios certificados. También necesita sus credenciales del portal de clientes de Red Hat para la autenticación. Pueden estar sujetos a suscripción o licencias a discreción del socio.

El registro Quay.io

Red Hat también administra el registro de contenedores Quay.io, donde cualquier persona puede registrarse para obtener una cuenta gratuita y publicar sus propias imágenes de contenedor.

Red Hat no ofrece garantías sobre ninguna imagen de contenedor alojada en Quay.io. Pueden variar desde experimentos de una sola vez sin nada de mantenimiento; pasando por imágenes de contenedores buenas, estables y con el debido mantenimiento de comunidades de código abierto sin acuerdo de nivel de servicio (SLA); hasta productos totalmente soportados de proveedores que pueden ofrecer acceso gratuito y sin autenticación a sus imágenes de contenedores para pruebas de productos.

La mayoría de los usuarios usan Quay.io como registro público, pero las organizaciones también pueden comprar planes que permiten usar Quay.io como registro privado.

Implementación de registros de contenedores empresariales

Acceder a registros externos, públicos o privados a través de Internet es muy conveniente, pero muchas organizaciones no permiten que los desarrolladores extraigan y ejecuten imágenes de contenedor externas. Estas organizaciones limitan a los desarrolladores a un conjunto de imágenes de contenedor que pasan criterios de seguridad, calidad y cumplimiento.

Confiar en los registros externos para extraer y enviar imágenes para sus hosts de producción no está exento de riesgos. Por ejemplo, cuando el registro está inactivo debido a una falla o a mantenimiento planificado por el proveedor, no puede implementar nuevos contenedores. En caso de falla, el escalamiento automático (autoscaling) de OpenShift también falla, porque OpenShift no puede extraer las imágenes que necesita para iniciar pods adicionales. Según su ancho de banda, extraer y enviar imágenes de Internet también puede ser un proceso lento.

En algunas organizaciones, las imágenes de contenedor son solo para uso interno y no se pueden hacer públicas. La configuración de un registro empresarial resuelve este problema. Después de establecer un registro empresarial, configure los hosts de contenedor dentro de la organización para extraer imágenes solo de este registro en vez de los registros externos predeterminados.

Al ejecutar un servidor de registro en su organización, puede mitigar los riesgos e implementar características adicionales. Por ejemplo, puede crear diferentes entornos y controlar quién puede extraer o enviar imágenes a ellos. Puede definir un flujo de trabajo de aprobación para mover las imágenes validadas del desarrollo a la producción. Puede implementar la exploración de vulnerabilidades y enviar una notificación cuando el escáner detecte una falla en una imagen en producción.

Software del servidor de registro

Entre el software de servidor de registro disponible se encuentran *Red Hat Quay Enterprise*, el servidor Docker-Distribution de código abierto y productos como JFrog y Nexus. OpenShift puede implementar contenedores desde cualquiera de estos servidores de registro.

Red Hat Quay Enterprise es un registro de imágenes de contenedor con funciones avanzadas, como el escaneo de seguridad de imágenes, el acceso basado en roles, la administración de organizaciones y equipos, la automatización de compilación de imágenes, la auditoría, la georreplicación y la alta disponibilidad.

Red Hat Quay Enterprise proporciona una interfaz web y una API REST. Se puede implementar como un contenedor local, en proveedores de nube seleccionados y también en *Red Hat OpenShift Container Platform*. También es el software de servidor detrás de *Quay.io*.

Si implementa *Quay Enterprise* en *OpenShift*, no reemplaza el registro interno del clúster. Una instancia de *Quay Enterprise* que se ejecuta en un clúster *OpenShift* es, a todos los efectos prácticos, como cualquier otra aplicación de *OpenShift*, y normalmente está disponible para otros clústeres de *OpenShift* y cualquier otro host de contenedor de la organización.

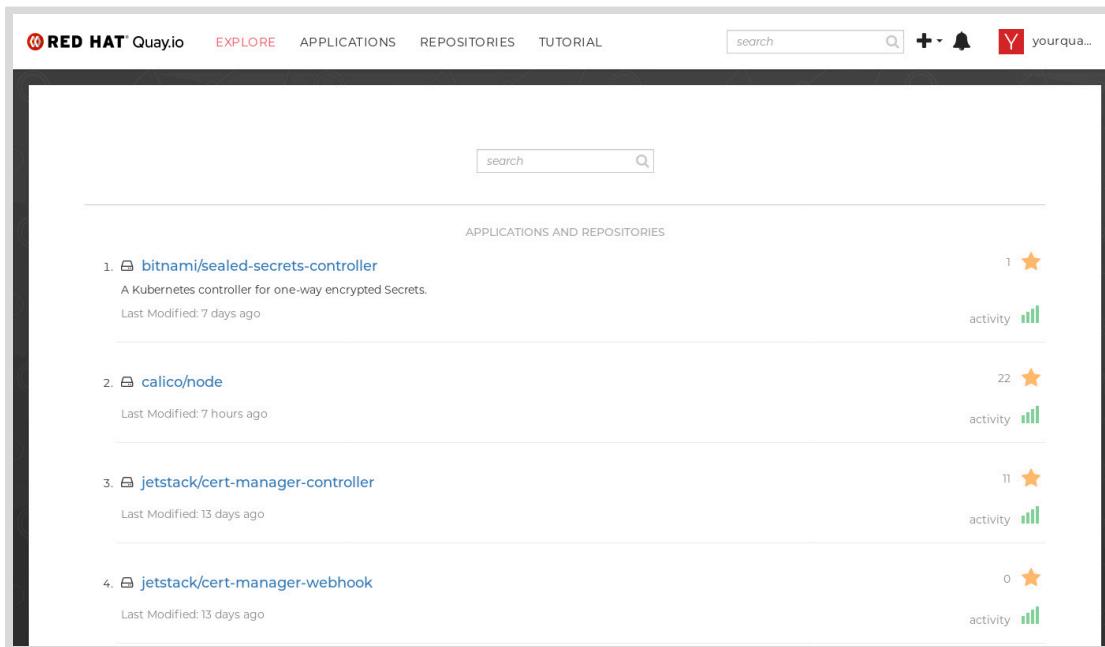


Figura 3.2: Página web de Quay.io

Acceso a registros de contenedor

El acceso a un registro público desde *OpenShift* normalmente no requiere ninguna configuración porque se supone que un registro público debe presentar un certificado TLS proporcionado por una entidad de certificación (CA) de confianza.

El acceso a un registro privado desde OpenShift normalmente requiere una configuración adicional para administrar las credenciales y los tokens de autenticación. Un registro empresarial también puede requerir la configuración de CA internas.

Para tener acceso a los registros de contenedor, use la *API de distribución de OCI*, que se basa en la API de Docker Registry anterior. Para usar la API, Red Hat recomienda usar las herramientas de contenedor de Red Hat Enterprise Linux (RHEL): Podman, Buildah y Skopeo. Las herramientas de contenedor de RHEL pueden personalizar, implementar y depurar imágenes de contenedor en función de los estándares de OCI.

La organización Open Container Initiative (OCI) define estándares abiertos para el formato de imágenes de contenedor, el tiempo de ejecución de contenedor y las API REST relacionadas. El formato de imágenes de contenedor OCI es una carpeta del sistema de archivos con archivos discretos que almacenan el manifiesto, los metadatos y las capas de las imágenes de contenedor.

Administración de contenedores con Podman

Podman es una herramienta diseñada para gestionar pods y contenedores sin necesidad de un daemon de contenedor, lo que reduce la superficie de ataque y mejora el rendimiento. Los procesos de pod y contenedor se crean como procesos secundarios del comando `podman`.

Podman puede reiniciar los contenedores detenidos y también insertar, confirmar, configurar, compilar y crear imágenes de contenedor. El comando `podman` suele seguir la sintaxis de comandos `docker` y proporciona capacidades adicionales, como la administración de pods. Podman no admite comandos `docker` que no estén relacionados con el motor de contenedores, como el modo de colmena.

Autenticación con registros

Para acceder a un registro privado, normalmente debe autenticarse. Podman proporciona el subcomando `login`, que genera un token de acceso y lo almacena para su reutilización posterior.

```
[user@host ~]$ podman login quay.io
Username: developer1
Password: MyS3cret!
Login Succeeded!
```

Después de la autenticación correcta, Podman almacena un token de acceso en el archivo `/run/user/UID/containers/auth.json`. El prefijo de ruta `/run/user/UID` no es fijo y proviene de la variable de entorno `XDG_RUNTIME_DIR`.

Puede iniciar sesión simultáneamente en varios registros con Podman. Cada inicio de sesión nuevo agrega o actualiza un token de acceso en el mismo archivo. El FQDN del servidor de registro indexa cada token de acceso.

Para cerrar sesión en un registro, use el subcomando `logout`:

```
[user@host ~]$ podman logout quay.io
Remove login credentials for registry.redhat.io
```

Para cerrar sesión en todos los registros y descartar todos los tokens de acceso que se almacenaron para su reutilización, use la opción `--all`:

```
[user@host ~]$ podman logout --all
Remove login credentials for all registries
```

Skopeo y Buildah también pueden usar los tokens de autenticación almacenados por Podman, pero no pueden presentar un mensaje de contraseña interactivo.

Podman requiere TLS y la verificación del certificado remoto de forma predeterminada. Si el servidor de registro no está configurado para usar TLS o está configurado para usar un certificado TLS autofirmado o un certificado TLS firmado por una CA desconocida, puede agregar la opción `--tls-verify=false` a los subcomandos `login` y `pull`.

Gestión de registros de contenedor con Skopeo

Red Hat admite el comando `skopeo` para gestionar imágenes en un registro de imágenes de contenedor. Skopeo no usa un motor de contenedores, por lo que es más eficiente que usar los subcomandos `tag`, `pull` y `push` de Podman.

Skopeo también proporciona capacidades adicionales que no se encuentran en Podman, como la firma y la eliminación de imágenes de contenedor de un servidor de registro.

El comando `skopeo` toma un subcomando, opciones y argumentos:

```
[user@host ~]$ skopeo subcommand [options] location...
```

Subcomandos principales

- `copy` para copiar imágenes de una ubicación a otra.
- `delete` para eliminar imágenes de un registro.
- `inspect` para ver los metadatos de una imagen.

Opciones principales

`--creds username:password`

Para ofrecer credenciales de inicio de sesión o un token de autenticación al registro.

`--[src-|dest-]tls-verify=false`

Desactiva la verificación de certificados TLS.

Para la autenticación en registros privados, Skopeo también puede usar el mismo archivo `auth.json` creado por el comando `podman login`. Como alternativa, puede pasar sus credenciales en la línea de comandos, como se muestra a continuación.

```
[user@host ~]$ skopeo inspect --creds developer1:MyS3cret! \
> docker://registry.redhat.io/rhscl/postgresql-96-rhel7
```

**Advertencia**

Aunque puede proporcionar credenciales a las herramientas de línea de comandos, esto crea una entrada en el historial de comandos, junto con otros problemas de seguridad. Use técnicas para evitar pasar credenciales de texto sin formato a los comandos:

```
[user@host ~]$ read -p "PASSWORD: " -s password
PASSWORD:
[user@host ~]$ skopeo inspect --creds developer1:$password \
> docker://registry.redhat.io/rhscl/postgresql-96-rhel7
```

Skopeo usa URI para representar ubicaciones de imágenes de contenedor y esquemas de URI para representar formatos de imagen de contenedor y API de registro. En la siguiente lista, se muestran los esquemas de URI más comunes:

oci

denota imágenes de contenedor almacenadas en una carpeta local con formato OCI.

docker

denota imágenes de contenedor remoto almacenadas en un servidor de registro.

containers-storage

denota las imágenes de contenedor almacenadas en la memoria caché del motor de contenedores local.

Enviar y etiquetar imágenes en un servidor de registro

El subcomando copy de Skopeo puede copiar imágenes de contenedor directamente entre registros, sin guardar las capas de imagen en el almacenamiento de contenedores local. También puede copiar imágenes de contenedor desde el motor de contenedores local a un servidor de registro y etiquetar estas imágenes en una sola operación.

Para copiar una imagen de contenedor denominada `myimage` del motor de contenedores local a un registro público inseguro en `registry.example.com` bajo la organización `myorg` o la cuenta de usuario:

```
[user@host ~]$ skopeo copy --dest-tls-verify=false \
> containers-storage:myimage \
> docker://registry.example.com/myorg/myimage
```

Para copiar una imagen de contenedor de la carpeta `/home/user/myimage` con formato OCI al registro público inseguro en `registry.example.com` bajo la organización o cuenta de usuario `myorg`:

```
[user@host ~]$ skopeo copy --dest-tls-verify=false \
> oci:/home/user/myimage \
> docker://registry.example.com/myorg/myimage
```

Al copiar imágenes de contenedor entre registros privados, puede autenticarse en ambos registros mediante Podman antes de invocar el subcomando `copy`, o puede usar las opciones `--src-creds` y `--dest-creds` para especificar las credenciales de autenticación, como se muestra a continuación:

```
[user@host ~]$ skopeo copy --src-creds=testuser:testpassword \
> --dest-creds=testuser1:testpassword \
> docker://srcregistry.domain.com/org1/private \
> docker://dstegistry.domain2.com/org2/private
```

Los argumentos del comando `skopeo` son siempre nombres de imagen completos. El siguiente ejemplo es un comando no válido porque ofrece solamente el nombre del servidor de registro como argumento de destino:

```
[user@host ~]$ skopeo copy oci:myimage \
> docker://registry.example.com/
```

El comando `copy` de `Skopeo` también puede etiquetar imágenes en repositorios remotos. En el ejemplo siguiente, se etiqueta una imagen existente con la etiqueta `1.0` como `latest`:

```
[user@host ~]$ skopeo copy docker://registry.example.com/myorg/myimage:1.0 \
> docker://registry.example.com/myorg/myimage:latest
```

Para alcanzar mayor eficiencia, `Skopeo` no lee ni envía capas de imagen que ya existen en el destino. Primero lee el manifiesto de la imagen de origen, luego determina qué capas ya existen en el destino y, a continuación, solo copia las que faltan. Si copia varias imágenes creadas a partir del mismo elemento primario, `Skopeo` no copia las capas primarias varias veces.

Eliminación de imágenes de un registro

Para eliminar la imagen de contenedor `myorg/myimage` del registro en `registry.example.com`, ejecute el siguiente comando:

```
[user@host ~]$ skopeo delete docker://registry.example.com/myorg/myimage
```

El subcomando `delete` puede adoptar, de manera opcional, las opciones `--creds` y `--tls-verify=false`.

Autenticación de OpenShift para registros privados

OpenShift también requiere credenciales para tener acceso a las imágenes de contenedor en registros privados. Estas credenciales se almacenan como secretos.

Puede proporcionar sus credenciales de registro privado directamente al comando `oc create secret`:

```
[user@host ~]$ oc create secret docker-registry registrycreds \
> --docker-server registry.example.com \
> --docker-username youruser \
> --docker-password yourpassword
```

Otra forma de crear el secreto es usar el token de autenticación del comando `podman login`:

```
[user@host ~]$ oc create secret generic registrycreds \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
```

Luego vincula el secreto con la cuenta de servicio default (predeterminada) de su proyecto:

```
[user@host ~]$ oc secrets link default registrycreds --for pull
```

Para usar el secreto para acceder a una imagen de compilador S2I, vincule el secreto a la cuenta de servicio builder (compilador) desde el proyecto:

```
[user@host ~]$ oc secrets link builder registrycreds
```



Referencias

Open Container Initiative (OCI)

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

Introducción práctica a la terminología de contenedores

<https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction>

Autenticación de registro de Red Hat Container

<https://access.redhat.com/RegistryAuthentication>

Encontrará más información sobre las herramientas de contenedor de RHEL en la guía *Compilación, ejecución y gestión de contenedores* para Red Hat Enterprise Linux 8; en

https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html-single/building_running_and_managing_containers/index

► Ejercicio Guiado

Uso de un registro empresarial

En este ejercicio, interactuará con un servidor de registro de imágenes de contenedor.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Enviar imágenes a un registro de contenedores autenticado externo.
- Implementar una aplicación contenedorizada en OpenShift mediante un registro de contenedores autenticado externo como entrada.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Los comandos `podman` y `skopeo`.
- Archivos que cumplen con OCI para la imagen de contenedor de prueba (`ubi-sleep`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

```
[student@workstation ~]$ lab external-registry start
```

- 1. Inicie sesión en un registro externo y envíe una imagen a este desde una carpeta compatible con OCI en el disco.
- 1.1. Inspeccione las capas de la imagen OCI de contenedor `ubi-sleep` en el disco local. Las imágenes OCI se almacenan como una carpeta de sistema de archivos que contiene varios archivos:

```
[student@workstation ~]$ ls ~/D0288/labs/external-registry/ubi-sleep
blobs  index.json  oci-layout
```

- 1.2. 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
```

- 1.3. Inicie sesión en su cuenta personal de Quay.io con Podman. Podman le solicita que introduzca su contraseña de Quay.io.

```
[student@workstation ~]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 1.4. Copie la imagen OCI en el registro externo de Quay.io con Skopeo y etiquétela como 1.0.

También puede ejecutar o cortar y pegar el comando `skopeo copy` siguiente del script `push-image.sh` en la carpeta `/home/student/D0288/labs/external-registry`.

```
[student@workstation ~]$ skopeo copy \
> oci:/home/student/D0288/labs/external-registry/ubi-sleep \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/ubi-sleep:1.0
...output omitted...
Writing manifest to image destination
Storing signatures
```

- 1.5. Compruebe que la imagen exista en el registro externo con Podman.



nota

Si no puede encontrar la imagen en los resultados de `podman search`, vaya al paso siguiente. Quay.io pueden truncar los resultados de búsqueda que devolverían demasiadas coincidencias.

```
[student@workstation ~]$ podman search quay.io/ubi-sleep
INDEX      NAME
...output omitted...
quay.io    quay.io/yourquayuser/ubi-sleep
...output omitted...
```

- 1.6. Inspeccione la imagen en el registro externo con Skopeo:

```
[student@workstation ~]$ skopeo inspect \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/ubi-sleep:1.0
{
  "Name": "quay.io/yourquayuser/ubi-sleep",
  "Tag": "1.0",
  ...output omitted...
```

- ▶ 2. Compruebe que Podman pueda ejecutar imágenes desde el registro externo.

- 2.1. Inicie un contenedor de prueba de la imagen en el registro externo. Debe iniciar sesión en Quay.io de nuevo porque ahora necesita ejecutar Podman como usuario root.

```
[student@workstation ~]$ sudo podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
[student@workstation ~]$ sudo podman run -d --name sleep \
> quay.io/${RHT_OCP4_QUAY_USER}/ubi-sleep:1.0
Trying to pull quay.io/youruser/ubi-sleep:1.0...Getting image source signatures
...output omitted...
```

2.2. Verifique que el nuevo contenedor esté ejecutándose:

```
[student@workstation ~]$ sudo podman ps
CONTAINER ID        IMAGE               ...   NAMES
63c5167376e5      quay.io/youruser/ubi-sleep:1.0 ...   sleep
```

2.3. Verifique que el nuevo contenedor genere una salida de registro:

```
[student@workstation ~]$ sudo podman logs sleep
...output omitted...
sleeping
sleeping
```

2.4. Detenga y elimine el contenedor de prueba:

```
[student@workstation ~]$ sudo podman stop sleep
...output omitted...
[student@workstation ~]$ sudo podman rm sleep
...output omitted...
```

► 3. Implemente una aplicación en OpenShift basada en la imagen del registro externo:

3.1. Inicie sesión en OpenShift y cree un proyecto nuevo. Coloque como prefijo del nombre del proyecto su nombre de usuario de desarrollador.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-external-registry
Now using project "youruser-docker-build" on server "https://
api.cluster.domain.example.com:6443".
```

3.2. Intente implementar una aplicación de la imagen de contenedor en el registro externo. Se producirá un error porque OpenShift necesita credenciales para acceder al registro externo.

```
[student@workstation ~]$ oc new-app --as-deployment-config --name sleep \
> --docker-image quay.io/${RHT_OCP4_QUAY_USER}/ubi-sleep:1.0
error: unable to locate any local docker images with name "quay.io/yourquayuser/
ubi-sleep:1.0"
...output omitted...
```

- 3.3. Cree un secreto con el token de acceso de la API de registro de contenedores almacenado por Podman.

También puede ejecutar o cortar y pegar el comando `oc create secret` siguiente del script `create-secret.sh` en la carpeta `/home/student/D0288/labs/external-registry`.

```
[student@workstation ~]$ oc create secret generic quayio \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
secret/quayio created
```

- 3.4. Vincule el secreto nuevo con la cuenta de servicio `default` (predeterminada).

```
[student@workstation ~]$ oc secrets link default quayio --for pull
...output omitted...
```

- 3.5. Implemente una aplicación de la imagen de contenedor en el registro externo. Esta vez OpenShift puede acceder al registro externo.

```
[student@workstation ~]$ oc new-app --as-deployment-config --name sleep \
> --docker-image quay.io/${RHT_OCP4_QUAY_USER}/ubi-sleep:1.0
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "sleep" created
deploymentconfig.apps.openshift.io "sleep" created
--> Success
...output omitted...
```

- 3.6. Espere hasta que el pod de aplicación esté listo y en ejecución:

```
[student@workstation ~]$ oc get pod
NAME        READY   STATUS    RESTARTS   AGE
sleep-1-deploy  1/1     Running   0          17s
sleep-1-mmtf8   1/1     Running   0          24s
```

- 3.7. Verifique que el pod genere una salida de registro:

```
[student@workstation ~]$ oc logs sleep-1-mmtf8
...output omitted...
sleeping
sleeping
```

- 4. Elimine el proyecto en OpenShift y el contenedor y la imagen en el registro de contenedores externo. Dado que Quay.io permite recuperar imágenes de contenedor antiguas, también debe eliminar el repositorio en Quay.io.

- 4.1. Elimine el proyecto de OpenShift:

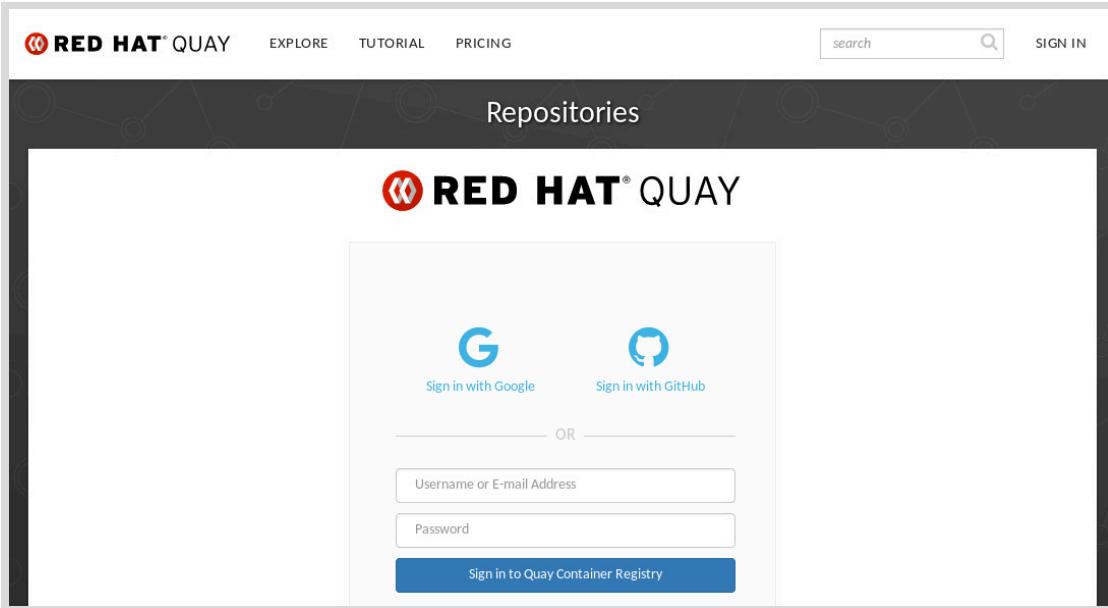
```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-external-registry
project.project.openshift.io "external-registry" deleted
```

- 4.2. Elimine la imagen de contenedor del registro externo:

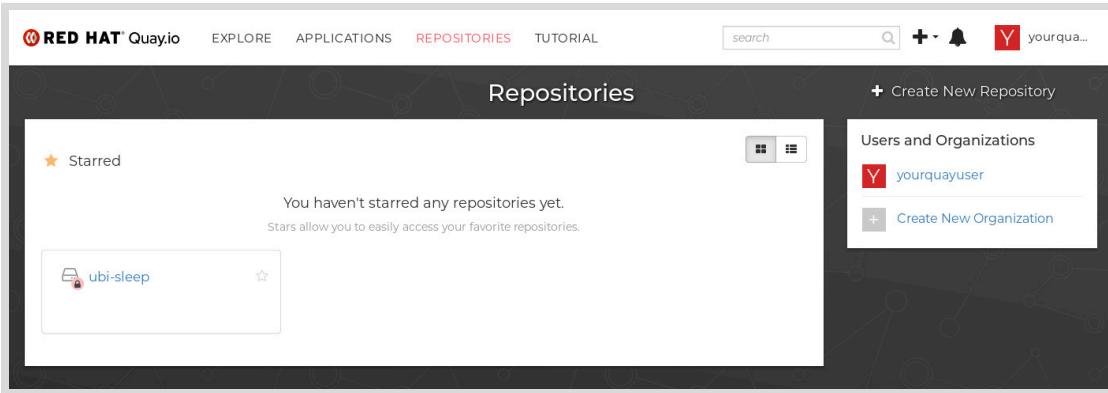
```
[student@workstation ~]$ skopeo delete \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/ubi-sleep:1.0
```

- 4.3. Inicie sesión en Quay.io con su cuenta personal gratuita.

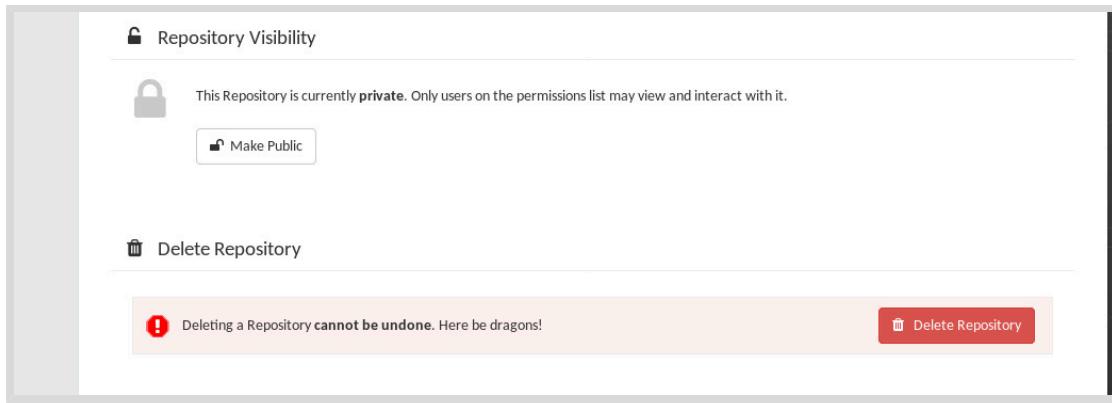
Desplácese hasta <http://quay.io> y haga clic en **Sign In** (Iniciar sesión) para proporcionar sus credenciales de usuario. Haga clic en **Sign in to Quay Container Registry** (Iniciar sesión en el registro de contenedor de Quay) para iniciar sesión en Quay.io.



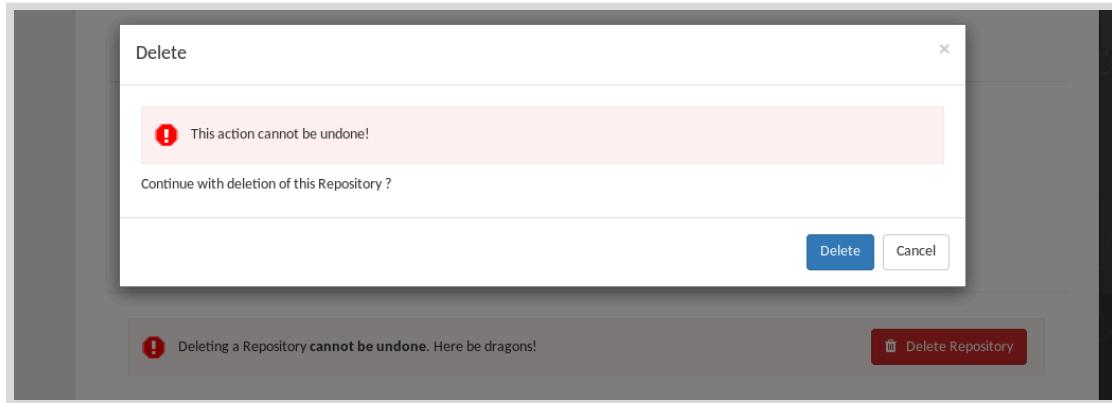
- 4.4. En el menú principal de Quay.io, haga clic en **Repositories** (Repositorios) y busque **ubi-sleep**. El icono de candado indica que es un repositorio privado que requiere autenticación para extracciones e inserciones. Haga clic en **ubi-sleep** para mostrar la página **Repository Activity** (Actividad del repositorio).



- 4.5. En la página **Repository Activity** (Actividad del repositorio) del repositorio **ubi-sleep**, desplácese hacia abajo y haga clic en el ícono de engranaje para mostrar la pestaña **Settings** (Configuración). Desplácese hacia abajo y haga clic en **Delete Repository** (Eliminar repositorio).



- 4.6. En el cuadro de diálogo Delete (Eliminar), haga clic en Delete para confirmar que desea eliminar el repositorio `ubi-sleep`. Después de unos momentos, volverá a la página **Repositories** (Repositorios). Ahora puede cerrar sesión en Quay.io.



Finalizar

En la máquina virtual `workstation`, ejecute el comando `lab external-registry finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab external-registry finish
```

Esto concluye el ejercicio guiado.

Permiso de acceso al registro de OpenShift

Objetivos

Tras finalizar esta sección, deberá ser capaz de acceder al registro interno de OpenShift con las herramientas de contenedor de Linux.

Revisión del registro interno

OpenShift ejecuta un servidor de registro interno para admitir flujos de trabajo de desarrollador basados en fuente a imagen (S2I). Los desarrolladores crean nuevas imágenes de contenedor con S2I mediante la creación de una configuración de compilación con el comando `oc new-app` y otros medios. Una configuración de compilación crea imágenes de contenedor a partir de código fuente o un Dockerfile y las almacena en el registro interno.

Los desarrolladores no están obligados a usar el registro interno. Pueden crear sus imágenes de contenedor localmente, con las herramientas de contenedor de Red Hat Enterprise Linux, y enviarlas a un registro público o privado externo al que OpenShift pueda acceder. También pueden ajustar sus configuraciones de compilación para enviar la imagen final directamente a un registro público o privado externo.

El instalador de OpenShift implementa un registro interno de forma predeterminada para que los desarrolladores puedan iniciar el trabajo de desarrollo tan pronto como el clúster OpenShift esté disponible para ellos. Sin un registro interno, los desarrolladores tendrían que esperar hasta que alguien implemente un servidor de registro y les proporcione credenciales de acceso. También tendrían que aprender a configurar secretos a fin de acceder a registros privados para poder realizar cualquier trabajo de desarrollo.

Hay algunos casos de uso para acceder a un registro interno de OpenShift fuera del proceso S2I. Por ejemplo:

- Una organización ya crea imágenes de contenedor localmente y aún no está lista para cambiar su flujo de trabajo de desarrollo. Estas organizaciones pueden tener un registro privado con características limitadas y desean reemplazarlo con el registro interno de OpenShift.
- Una organización mantiene varios clústeres de OpenShift y necesita copiar imágenes de contenedor de un desarrollo a un clúster de producción. Estas organizaciones pueden tener una herramienta de CI/CD que promueve imágenes de un registro interno a un registro externo u otro registro interno.
- Un proveedor de software independiente (ISV) crea imágenes de contenedor para sus clientes y las publica en un registro privado mantenido por un proveedor de servicios en la nube, como Quay.io, o en un registro empresarial que mantiene el ISV.

El registro interno de OpenShift puede proporcionar todas las funciones que requiere un cliente, como controles de acceso detallados basados en usuarios, grupos y roles de OpenShift. El registro interno puede proporcionar mejores funciones o una mayor facilidad de uso en comparación con el registro empresarial actual de una organización, por ejemplo, si se basa en el servidor de registro de Docker-Distribution. Estas organizaciones pueden eliminar gradualmente su registro actual y usar el registro interno de OpenShift como su nuevo registro empresarial.

Otros clientes pueden requerir funciones avanzadas, como el análisis de seguridad de imágenes y la replicación geográfica, y adoptar un servidor de registro empresarial más potente, como Red Hat Quay Enterprise. Los clientes que adoptan un registro empresarial más eficaz todavía pueden necesitar exponer el registro interno para poder copiar imágenes en el registro empresarial de la organización.

El operador de registro de imágenes

El instalador de OpenShift configura el registro interno para que solo sea accesible desde el interior de su clúster OpenShift. Exponer el registro interno para el acceso externo es un procedimiento sencillo, pero requiere privilegios de administración del clúster.

El *operador de registro de imágenes* de OpenShift administra el registro interno. Todas las opciones de configuración del operador de registro de imágenes se encuentran en el recurso de configuración del clúster en el proyecto `openshift-image-registry`. Cambie el atributo `spec.defaultRoute` a `true` y el operador de registro de imágenes creará una ruta para exponer el registro interno. Una forma de realizar ese cambio es con el siguiente comando `oc patch`:

```
[user@host ~] oc patch config cluster -n openshift-image-registry \
> --type merge -p '{"spec":{"defaultRoute":true}}'
```

La ruta `default-route` usa el nombre de dominio comodín predeterminado para la aplicación implementada en el clúster:

```
[user@host ~] oc get route -n openshift-image-registry
NAME          HOST/PORT
default-route default-route-openshift-image-registry.domain.example.com ...
```

Autenticación en un registro interno

Para iniciar sesión en un registro interno con las herramientas de contenedor de Linux, debe capturar el token de autenticación de OpenShift del usuario.

Use el comando `oc whoami -t` para capturar el token. El token es una larga cadena aleatoria. Es más fácil escribir comandos si guarda el token como una variable de shell:

```
[user@host ~] TOKEN=$(oc whoami -t)
```

Use el token como parte de un subcomando `login` de Podman:

```
[user@host ~] podman login -u myuser -p ${TOKEN} \
> default-route-openshift-image-registry.domain.example.com
```

También puede usar el token como valor de las opciones `--[src|dst]creds` de Skopeo.

```
[user@host ~] skopeo inspect --creds=myuser:${TOKEN} \
> docker://default-route-openshift-image-registry.domain.example.com/...
```

Acceso al registro interno como registro seguro o inseguro

Si el clúster OpenShift está configurado con un certificado TLS válido para su dominio comodín, puede usar las herramientas de contenedor de Linux para trabajar con imágenes dentro de cualquier proyecto al que tenga acceso.

En el ejemplo siguiente, se usa Skopeo para inspeccionar la imagen de contenedor de la aplicación myapp dentro del proyecto `myproj`. Se supone que un podman login anterior se realizó correctamente.

```
[user@host ~] skopeo inspect \
> docker://default-route-openshift-image-registry.domain.example.com/myproj/myapp
```

Si el clúster OpenShift usa la entidad de certificación (CA) que el instalador de OpenShift genera de forma predeterminada, debe tener acceso al registro interno como un registro inseguro:

```
[user@host ~] skopeo inspect --tls-verify=false \
> docker://default-route-openshift-image-registry.domain.example.com/myproj/myapp
```

Un administrador de clúster puede configurar la ruta para el registro interno de diferentes maneras, por ejemplo, mediante una CA interna mantenida por su organización. En este escenario, la estación de trabajo para desarrolladores puede o no estar configurada para confiar en el certificado TLS del registro interno.

Su organización también puede recuperar el certificado público de la CA interna de su clúster OpenShift y declararlo de confianza dentro de su organización. El administrador de clúster puede configurar una ruta alternativa, con un nombre de host más corto, para exponer el registro interno.

Estos escenarios están fuera del ámbito de este curso. Consulte los cursos de capacitación de Red Hat sobre la serie de administración de OpenShift, como *Red Hat OpenShift Administration I* (DO280) y *Red Hat Security: Securing Containers and OpenShift* (DO425), para obtener más información sobre la configuración de certificados TLS para OpenShift y el motor de contenedores local.

Concesión de acceso a las imágenes en un registro interno

Cualquier usuario con acceso a un proyecto de OpenShift puede extraer o enviar imágenes hacia y desde ese proyecto, según su nivel de acceso. Si un usuario tiene los roles `admin` o `edit` en el proyecto, puede extraer o enviar imágenes de ese proyecto. Si en su lugar solo tiene el rol `view` en el proyecto, solo pueden extraer imágenes de ese proyecto.

OpenShift también ofrece algunos roles especializados para cuando desea conceder acceso solo a imágenes dentro de un proyecto y no conceder acceso para realizar otras tareas de desarrollo, como la creación e implementación de aplicaciones dentro del proyecto. A continuación, se detallan los roles más comunes:

`registry-viewer` y `system:image-puller`

Estos roles permiten que un usuario extraiga e inspeccione imágenes del registro interno.

`registry-editor` y `system:image-pusher`

Estos roles permiten que un usuario envíe y etiquete imágenes del registro interno.

Los roles **system:*** proporcionan las capacidades mínimas necesarias para extraer y enviar imágenes del registro interno. Como se indicó anteriormente, los usuarios de OpenShift que ya tienen roles de **admin** o **edit** en un proyecto no necesitan estos roles **system:***.

Los roles **registry-*** proporcionan capacidades más completas en torno a la administración de registros para las organizaciones que desean usar el registro interno como su registro empresarial. Estos roles conceden derechos adicionales, como la creación de nuevos proyectos, pero no conceden otros derechos, como la creación e implementación de aplicaciones. Los estándares OCI no especifican cómo administrar un registro de imágenes, por lo que quien administre un registro interno de OpenShift debe conocer los conceptos de administración de OpenShift y el comando `oc`. Esto hace que el **registry-*** sea menos útil.

En el ejemplo siguiente, se permite que un usuario extraiga imágenes del registro interno en un proyecto determinado. Debe tener acceso de administrador a todo el proyecto o a todo el clúster para usar el comando `oc policy`.

```
[user@host ~] oc policy add-role-to-user system:image-puller \
> user_name -n project_name
```

Los conceptos generales de autenticación y autorización de OpenShift están fuera del alcance de este curso. Consulte los cursos de capacitación de Red Hat sobre la serie de administración de OpenShift, como *Red Hat OpenShift Administration I (DO280)* y *Red Hat Security: Securing Containers and OpenShift (DO425)*, para obtener más información sobre la configuración de certificados TLS para OpenShift y el motor de contenedores local.



Referencias

Para obtener más información sobre cómo exponer el registro interno, consulte el capítulo *Operador del registro de imágenes en OpenShift Container Platform* de la guía *Registro para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/registry/index#configuring-registry-operator

Para obtener más información sobre cómo conceder acceso a las imágenes del registro interno, consulte el capítulo *Acceso al registro* de la guía *Registro para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/registry/index#accessing-the-registry

► Ejercicio Guiado

Uso del registro de OpenShift

En este ejercicio, accederá al registro interno de OpenShift con las herramientas de contenedor de Linux.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Enviar una imagen a un registro interno con las herramientas de contenedor de Linux.
- Crear un contenedor a partir de un registro interno con las herramientas de contenedor de Linux.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución con su registro interno expuesto.
- Los comandos `podman` y `skopeo`.
- Archivos que cumplen con OCI para la imagen de contenedor de prueba `ubi-info`.

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

```
[student@workstation ~]$ lab expose-registry start
```

- 1. Compruebe que el registro interno del clúster OpenShift esté expuesto.

- 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 OpenShift con su cuenta de usuario de desarrollador.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

- 1.3. Compruebe que haya una ruta en el proyecto `openshift-image-registry`. La salida del comando `oc route` se editó para mostrar cada columna en una sola línea con fines de legibilidad porque se espera que el nombre de host sea demasiado largo para ajustarse al ancho del papel.

```
[student@workstation ~]$ oc get route -n openshift-image-registry
NAME          HOST/PORT
...output omitted...
default-route  default-route-openshift-image-
registry.apps.cluster.domain.example.com
...output omitted...
```

**nota**

Una instalación predeterminada de Red Hat OpenShift Container Platform no permite que un usuario normal vea ningún recurso en los proyectos `openshift-*`. El clúster de esta aula asigna a todas las cuentas de usuario de desarrollador de los estudiantes derechos adicionales para que realicen la operación anterior.

- 1.4. Para facilitar la escritura, guarde el nombre de host de la ruta del registro interno en una variable de shell.

Puede cortar y pegar el comando `oc get` siguiente del script `push-image.sh` en la carpeta `/home/student/D0288/labs/expose-registry`.

```
[student@workstation ~]$ INTERNAL_REGISTRY=$( oc get route default-route \
> -n openshift-image-registry -o jsonpath='{.spec.host}' )
```

- 1.5. Compruebe el valor de la variable de shell `INTERNAL_REGISTRY`. Debe coincidir con la salida del primer comando `oc get route`.

```
[student@workstation ~]$ echo ${INTERNAL_REGISTRY}
default-route-openshift-image-registry.apps.cluster.domain.example.com
```

- 2. Envíe una imagen al registro interno de OpenShift usando su cuenta de usuario de desarrollador.

- 2.1. Cree un proyecto para alojar los flujos de imágenes que gestionan las imágenes de contenedor que enviará al registro interno de OpenShift:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-common
Now using project "youruser-common" on server
"https://api.cluster.domain.example.com:6443".
```

- 2.2. Obtenga el token de autenticación de OpenShift de su cuenta de usuario de desarrollador para usar en comandos posteriores:

```
[student@workstation ~]$ TOKEN=$(oc whoami -t)
```

- 2.3. Verifique que la carpeta `ubi-info` contenga una imagen de contenedor con formato OCI:

```
[student@workstation ~]$ ls ~/D0288/labs/expose-registry/ubi-info
blobs  index.json  oci-layout
```

- 2.4. Copie la imagen de OCI en el registro interno del clúster del aula con Skopeo y etiquétela como 1.0. Use el nombre de host y el token recuperados en los pasos anteriores.

Puede cortar y pegar el comando `skopeo copy` siguiente del script `push-image.sh` en la carpeta `/home/student/D0288/labs/expose-registry`.

```
[student@workstation ~]$ skopeo copy \
> --dest-creds=${RHT_OCP4_DEV_USER}:${TOKEN} \
> oci:/home/student/D0288/labs/expose-registry/ubi-info \
> docker://${INTERNAL_REGISTRY}/${RHT_OCP4_DEV_USER}-common/ubi-info:1.0
...output omitted...
Writing manifest to image destination
Storing signatures
```

- 2.5. Verifique que se haya creado un flujo de imágenes para gestionar la nueva imagen de contenedor. La salida del comando `oc get is` se editó para mostrar cada columna en una sola línea con fines de legibilidad porque se espera que el nombre del repositorio de imágenes sea demasiado largo para ajustarse al ancho del papel.

```
[student@workstation ~]$ oc get is
NAME           IMAGE REPOSITORY
ubi-info      default-route-openshift-image-registry.apps....output omitted...
```

► 3. Cree un contenedor local a partir de la imagen del registro interno de OpenShift.

- 3.1. Inicie sesión en el registro interno de OpenShift con el token de autenticación de *Paso 2.2* y el nombre de host de registro de *Paso 1.4*:

```
[student@workstation ~]$ sudo podman login -u ${RHT_OCP4_DEV_USER} \
> -p ${TOKEN} ${INTERNAL_REGISTRY}
Login Succeeded!
```

- 3.2. Descargue la imagen de contenedor `ubi-info:1.0` en el motor de contenedores local.

```
[student@workstation ~]$ sudo podman pull \
> ${INTERNAL_REGISTRY}/${RHT_OCP4_DEV_USER}-common/ubi-info:1.0
...output omitted...
Writing manifest to image destination
Storing signatures
...output omitted...
```

- 3.3. Inicie un nuevo contenedor desde la imagen de contenedor `ubi-info:1.0`. En el contenedor se muestra información del sistema, como el nombre del host y la memoria libre; luego, se sale. En la siguiente salida, se omite información específica para el contenedor en ejecución:

```
[student@workstation ~]$ sudo podman run --name info \
> ${INTERNAL_REGISTRY}/${RHT_OCP4_DEV_USER}-common/ubi-info:1.0
...output omitted...
--- Host name:
...output omitted...
```

```
--- Free memory  
...output omitted...  
--- Mounted file systems (partial)  
...output omitted...
```

► **4.** Realice la limpieza. Elimine todos los recursos creados durante este ejercicio.

- 4.1. Elimine la imagen de contenedor del registro interno del clúster del aula eliminando su flujo de imágenes:

```
[student@workstation ~]$ oc delete is ubi-info  
imagestream.image.openshift.io "ubi-info" deleted
```

- 4.2. Elimine el proyecto de OpenShift:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-common  
project.project.openshift.io "youruser-common" deleted
```

- 4.3. Elimine el contenedor de prueba y la imagen del motor de contenedores local:

```
[student@workstation ~]$ sudo podman rm info  
...output omitted...  
[student@workstation ~]$ sudo podman rmi -f \  
> ${INTERNAL_REGISTRY}/${RHT_OCP4_DEV_USER}-common/ubi-info:1.0  
...output omitted...
```

Finalizar

En la máquina virtual `workstation`, ejecute el comando `lab expose-registry finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab expose-registry finish
```

Esto concluye el ejercicio guiado.

Creación de flujos de imágenes

Objetivos

Después de completar esta sección, deberá ser capaz de crear flujos de imágenes para imágenes de contenedor en registros externos.

Descripción de flujos de imágenes

Los *flujos de imágenes* son uno de los principales diferenciadores entre OpenShift y Kubernetes upstream. Los recursos de Kubernetes hacen referencia directamente a imágenes de contenedor, pero los recursos de OpenShift, como las configuraciones de implementación y las configuraciones de compilación, hacen referencia a flujos de imágenes. OpenShift también extiende los recursos de Kubernetes, como los recursos de StatefulSet y CronJob, con anotaciones que los hacen funcionar con flujos de imágenes de OpenShift.

Los flujos de imágenes permiten que OpenShift garantice implementaciones reproducibles y estables de aplicaciones contenedorizadas y también reverisiones de implementaciones al último estado bueno conocido.

Los flujos de imágenes proporcionan un nombre corto y estable para hacer referencia a una imagen de contenedor que es independiente de cualquier configuración de servidor de registro y tiempo de ejecución de contenedor.

Por ejemplo, una organización podría comenzar descargando imágenes de contenedor directamente desde el registro público de Red Hat y más tarde configurar un registro empresarial como duplicado de esas imágenes para ahorrar ancho de banda. Los usuarios de OpenShift no notarán ningún cambio porque todavía hacen referencia a estas imágenes con el mismo nombre de flujo de imágenes. Los usuarios de las herramientas de contenedor de RHEL notarán el cambio porque deberían cambiar los nombres de registro en sus comandos o cambiar sus configuraciones de motor de contenedores para buscar primero el duplicado local.

Hay otros escenarios en los que el direccionamiento indirecto proporcionado por un flujo de imágenes resulta ser útil. Supongamos que comienza con una imagen de contenedor de base de datos que tiene problemas de seguridad y el proveedor tarda demasiado tiempo en actualizar la imagen con correcciones. Más adelante encontrará un proveedor alternativo que proporciona una imagen de contenedor alternativa para la misma base de datos, con esos problemas de seguridad ya corregidos, y aún mejor, con un historial de proporcionar actualizaciones oportunas a ellos. Si esas imágenes de contenedor son compatibles con la configuración de variables y volúmenes de entorno, simplemente puede cambiar el flujo de imágenes para que apunte a la imagen del proveedor alternativo.

Red Hat proporciona imágenes de contenedor reforzadas y soportadas que funcionan principalmente como reemplazos directos de imágenes de contenedor de algunos proyectos de código abierto populares, como la base de datos MariaDB.

Descripción de etiquetas de flujo de imágenes

Un flujo de imágenes representa uno o varios conjuntos de imágenes de contenedor. Cada conjunto o flujo se identifica mediante una *etiqueta de flujo de imágenes*. A diferencia de las imágenes de contenedor de un servidor de registro, que tienen varias etiquetas del mismo

repositorio de imágenes (o usuario u organización), un flujo de imágenes puede tener varias etiquetas de flujo de imágenes que hacen referencia a imágenes de contenedor de diferentes servidores de registro y de diferentes repositorios de imágenes.

Un flujo de imágenes proporciona configuraciones predeterminadas para un conjunto de etiquetas de flujo de imágenes. Cada etiqueta de flujo de imágenes hace referencia a un flujo de imágenes de contenedor y puede invalidar la mayoría de las configuraciones de su flujo de imágenes asociado.

Una etiqueta de flujo de imágenes almacena una copia de los metadatos sobre su imagen de contenedor actual y, de manera opcional, puede almacenar una copia de las capas de imagen de contenedor actuales y anteriores. El almacenamiento de metadatos acelera la búsqueda y la inspección de imágenes de contenedor, ya que no es necesario llegar a su servidor de registro de origen.

El almacenamiento de capas de imagen permite que una etiqueta de flujo de imágenes actúe como caché de imágenes local, lo que evita la necesidad de capturar estas capas de su servidor de registro de origen. La etiqueta de flujo de imágenes almacena las capas de imágenes almacenadas en caché en el registro interno de OpenShift. Los consumidores de la imagen almacenada en caché, como los pods y las configuraciones de implementación, solo hacen referencia al registro interno como el registro de origen de la imagen.

Hay algunos otros tipos de recursos de OpenShift relacionados con flujos de imágenes, pero un desarrollador normalmente puede descartarlos como detalles de implementación del registro interno y ocuparse solo de los flujos de imágenes y las etiquetas de flujo de imágenes.

Para visualizar mejor la relación entre los flujos de imágenes y las etiquetas de flujo de imágenes, puede explorar el proyecto `openshift` que se ha creado previamente en todos los clústeres de OpenShift. Puede ver que hay una serie de flujos de imágenes en ese proyecto, incluido el flujo de imágenes `php`:

```
[user@host ~]$ oc get is -n openshift -o name
...output omitted...
imagestream.image.openshift.io/nodejs
imagestream.image.openshift.io/perl
imagestream.image.openshift.io/php
imagestream.image.openshift.io/postgresql
imagestream.image.openshift.io/python
...output omitted...
```

Hay un número de etiquetas para el flujo de imágenes `php`, y hay un recurso de flujo de imágenes para cada una:

```
[user@host ~]$ oc get istag -n openshift | grep php
php:7.2      image-registry...output omitted...   6 days ago
php:7.3      image-registry...output omitted...   6 days ago
php:latest   image-registry...output omitted...   6 days ago
```

Si usa el comando `oc describe` en un flujo de imágenes, muestra la información del flujo de imágenes y sus etiquetas de flujo de imágenes:

```
[user@host ~]$ oc describe is php -n openshift
Name:                  php
Namespace:             openshift
```

```

...output omitted...
Tags: 3

7.3 (latest)
tagged from registry.redhat.io/rhscl/php-73-rhel7:latest
...output omitted...
7.2
tagged from registry.redhat.io/rhscl/php-72-rhel7:latest
...output omitted...

```

En el ejemplo anterior, cada una de las etiquetas de flujo de imágenes php hace referencia a un nombre de imagen diferente.

Descripción de nombres, etiquetas e identificadores de imagen

El nombre textual de una imagen de contenedor es simplemente una cadena. Este nombre a veces se interpreta como que está formado por varios componentes, como `registry-host-name/repository-or-organization-or-user-name/image-name:tag-name`, pero dividir el nombre de la imagen en sus componentes es solo una cuestión de convención, no de estructura.

Un identificador de imagen identifica de forma única una imagen de contenedor inmutable mediante un hash SHA-256. Recuerde que no puede modificar una imagen de contenedor. En su lugar, debe crear una nueva imagen de contenedor que tenga un nuevo identificador. Al enviar una nueva imagen de contenedor a un servidor de registro, el servidor asocia el nombre textual existente con el nuevo identificador de imagen.

Cuando se inicia un contenedor desde un nombre de imagen, se descarga la imagen que está asociada actualmente a ese nombre de imagen. El ID de imagen real detrás de ese nombre puede cambiar en cualquier momento, y el siguiente contenedor que inicie puede tener un ID de imagen diferente. Si la imagen asociada a un nombre de imagen tiene algún problema y solo conoce el nombre de la imagen, no puede revertir a una imagen anterior.

Las etiquetas de flujo de imágenes de OpenShift mantienen un historial de los últimos identificadores de imagen que recuperaron de un servidor de registro. El historial de los identificadores de imagen es el *flujo* de imágenes de una etiqueta de flujo de imágenes. Puede usar el historial dentro de una etiqueta de flujo de imágenes para revertir a una imagen anterior, si, por ejemplo, una nueva imagen de contenedor produce un error de implementación.

La actualización de una imagen de contenedor en un registro externo no actualiza automáticamente una etiqueta de flujo de imágenes. La etiqueta de flujo de imágenes mantiene la referencia al último ID de imagen que obtuvo. Este comportamiento es crucial para ampliar las aplicaciones porque aísla a OpenShift de los cambios que se producen en un servidor de registro.

Supongamos que implementa una aplicación desde un registro externo y, después de unos días de pruebas con algunos usuarios, decide ampliar su implementación para habilitar un mayor cantidad de usuarios. Mientras tanto, el proveedor actualiza la imagen de contenedor en el registro externo. Si OpenShift no tuviera etiquetas de flujo de imágenes, los nuevos pods obtendrían la nueva imagen de contenedor, que es diferente de la imagen del pod original. En función de los cambios, esto podría provocar un error en la aplicación. Dado que OpenShift almacena el ID de imagen de la imagen original en una etiqueta de flujo de imágenes, puede crear nuevos pods con el mismo ID de imagen y evitar cualquier incompatibilidad entre la imagen original y la actualizada.

OpenShift mantiene el ID de imagen usado para el primer pod y garantiza que los pods nuevos usen el mismo ID de imagen. OpenShift garantiza que todos los pods usen exactamente la misma imagen.

Para visualizar mejor la relación entre un flujo de imágenes, una etiqueta de flujo de imágenes, un nombre de imagen y un ID de imagen, consulte el siguiente comando `oc describe is`, que muestra la imagen de origen y el ID de imagen actual para cada etiqueta de flujo de imágenes:

```
[user@host ~]$ oc describe is php -n openshift
Name:                  php
Namespace:             openshift
...output omitted...
7.3 (latest)
  tagged from registry.redhat.io/rhscl/php-73-rhel7:latest
  ...output omitted...
  * registry.redhat.io/rhscl/php-73-rhel7@sha256:22ba...09b5
  ...output omitted...
7.2
  tagged from registry.redhat.io/rhscl/php-72-rhel7:latest
  ...output omitted...
  * registry.redhat.io/rhscl/php-72-rhel7@sha256:e8d6...e615
  ...output omitted...
```

Si el administrador del clúster OpenShift ya actualizó la etiqueta de flujo de imágenes `php:7.3`, el comando `oc describe is` muestra varios identificadores de imagen para esa etiqueta:

```
[user@host ~]$ oc describe is php -n openshift
Name:                  php
Namespace:             openshift
...output omitted...
7.3 (latest)
  tagged from registry.redhat.io/rhscl/php-73-rhel7:latest
  ...output omitted...
  * registry.redhat.io/rhscl/php-73-rhel7@sha256:22ba...09b5
  ...output omitted...
  registry.redhat.io/rhscl/php-73-rhel7@sha256:bc61...1e91
  ...output omitted...
7.2
  tagged from registry.redhat.io/rhscl/php-72-rhel7:latest
  ...output omitted...
  * registry.redhat.io/rhscl/php-72-rhel7@sha256:e8d6...e615
  ...output omitted...
```

En el ejemplo anterior, el asterisco (*) muestra qué ID de imagen es el actual para cada etiqueta de flujo de imágenes. Por lo general, es el último en importarse y el primero en la lista.

Cuando una etiqueta de flujo de imágenes de OpenShift hace referencia a una imagen de contenedor de un registro externo, debe actualizar explícitamente la etiqueta de flujo de imágenes para obtener nuevos identificadores de imagen del registro externo. De forma predeterminada, OpenShift no monitorea los registros externos en busca de cambios en el identificador de imagen asociado a un nombre de imagen.

Puede configurar una etiqueta de flujo de imágenes para comprobar el registro externo en busca de actualizaciones en una programación definida. De forma predeterminada, las nuevas etiquetas de flujo de imágenes no comprueban si hay imágenes actualizadas.

Las configuraciones de compilación actualizan automáticamente la etiqueta de flujo de imágenes que usan como imagen de salida. Esto obliga a la redistribución de los pods de aplicación con esa etiqueta de flujo de imágenes.

Administración de flujos y etiquetas de imágenes

Para crear un recurso de etiquetas de flujo de imágenes para una imagen de contenedor hospedada en un registro externo, use el comando `oc import-image` con las opciones `--confirm` y `--from`. El siguiente comando actualiza una etiqueta de flujo de imágenes o crea una si no existe:

```
[user@host ~]$ oc import-image myimagestream[:tag] --confirm \
> --from registry/myorg/myimage[:tag]
```

Si no especifica un nombre de etiqueta, la etiqueta `latest` (más reciente) se usa de forma predeterminada. En este ejemplo, la etiqueta de flujo de imágenes hace referencia al flujo de imágenes `myimagestream`. Si el flujo de imágenes correspondiente aún no existe, OpenShift lo crea.



nota

Para crear un flujo de imágenes para imágenes de contenedor hospedadas en un servidor de registro que no esté configurado con un certificado TLS de confianza, agregue la opción `--insecure` al comando `oc import-image`.

El nombre de etiqueta de la etiqueta de flujo de imágenes puede ser diferente de la etiqueta de imagen de contenedor del servidor de registro de origen. En el ejemplo siguiente, se crea una etiqueta de flujo de imágenes `1.0` a partir de la etiqueta `latest` del servidor de registro de origen (por omisión):

```
[user@host ~]$ oc import-image myimagestream:1.0 --confirm \
> --from registry/myorg/myimage
```

Para crear un recurso de etiqueta de flujo de imágenes para cada etiqueta de imagen de contenedor que exista en el servidor de registro de origen, agregue la opción `--all` al comando `oc import-image`:

```
[user@host ~]$ oc import-image myimagestream --confirm --all \
> --from registry/myorg/myimage
```

Puede ejecutar el comando `oc import-image` en un flujo de imágenes existente para actualizar una de sus etiquetas de flujo de imágenes actuales con los identificadores de imagen actuales del servidor de registro de origen.

```
[user@host ~]$ oc import-image myimagestream[:tag] --confirm
```

La actualización de un flujo de imágenes con la opción `--all` actualiza todas las etiquetas de flujo de imágenes y también crea nuevas etiquetas de flujo de imágenes para las nuevas etiquetas que encuentra en el servidor de registro de origen.

Puede ejercer un control más preciso sobre una etiqueta de flujo de imágenes con el comando `oc tag`. Permite cambios tales como:

- Asociar una etiqueta de flujo de imágenes a un servidor de registro diferente al servidor asociado con su flujo de imágenes.
- Asociar una etiqueta de flujo de imágenes a un nombre y una etiqueta de imagen de contenedor diferentes.
- Asociar una etiqueta de flujo de imágenes a un identificador de imagen determinado, que puede no ser el asociado actualmente a esa etiqueta de imagen del servidor de registro.
- Asociar un flujo de imágenes con otra etiqueta de flujo de imágenes. Por ejemplo, el ejemplo anterior del comando `oc describe is` muestra que la etiqueta de flujo de imágenes `php:latest` sigue a la etiqueta de flujo de imágenes `php:7.3`. Esta es una forma de crear alias para etiquetas de flujo de imágenes.

Está fuera del alcance de este curso enseñar todos los aspectos de la gestión de flujo de imágenes, aunque algunos de ellos, como los eventos de cambio de imagen, se exploran en capítulos posteriores.

Uso de flujos de imágenes con registros privados

Para crear flujos de imágenes y etiquetas de flujo de imágenes que hagan referencia a un registro privado, OpenShift necesita un token de acceso a ese servidor de registro.

Proporcione ese token de acceso como secreto, de la misma manera que implementaría una aplicación desde un registro privado, y no tendrá que vincular el secreto a ninguna cuenta de servicio. El comando `oc import-image` busca en los secretos del proyecto actual uno que coincida con el nombre de host de registro.

En el ejemplo siguiente, se usa Podman para iniciar sesión en un registro privado, crear un secreto para almacenar el token de acceso y, a continuación, crear un flujo de imágenes que apunte al registro privado:

```
[user@host ~]$ podman login -u myuser registry.example.com
[user@host ~]$ oc create secret generic regtoken \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
[user@host ~]$ oc import-image myis --confirm \
> --from registry.example.com/myorg/myimage
```

Después de crear un flujo de imágenes, puede usarla para implementar una aplicación con el comando `oc new-app --as-deployment-config -i myis`. También puede usar ese flujo de imágenes como imagen de compilador en el comando `oc new-app --as-deployment-config myis~giturl`.

De forma predeterminada, un recurso de flujo de imágenes solo está disponible para crear aplicaciones o compilaciones en el mismo proyecto.

Compartir un flujo de imágenes entre varios proyectos

Es una práctica común crear proyectos en OpenShift para almacenar recursos que se comparten entre varios usuarios y equipos de desarrollo. Estos proyectos compartidos almacenan recursos como flujos de imágenes y plantillas, a las que los desarrolladores hacen referencia al implementar aplicaciones en sus proyectos.

OpenShift viene con un proyecto compartido llamado `openshift`, que proporciona plantillas de aplicación de inicio rápido y también secuencias de imágenes para compiladores S2I para lenguajes de programación populares como Python y Ruby. Algunas organizaciones crean

proyectos similares para sus equipos en lugar de agregar recursos al proyecto openshift, a fin de evitar problemas al actualizar el clúster a una nueva versión de OpenShift.



Importante

Tenía la opción de agregar nuevos recursos al proyecto openshift en versiones anteriores, pero desde Red Hat OpenShift Container Platform 4, el operador Samples (Ejemplos) administra el proyecto openshift y puede eliminar recursos que agregue manualmente en cualquier momento.

Para compilar e implementar aplicaciones mediante un flujo de imágenes definida en otro proyecto, tiene dos opciones:

- Cree un secreto con un token de acceso al registro privado en cada proyecto que use el flujo de imágenes y vincule ese secreto a las cuentas de servicio de cada proyecto.
- Cree un secreto con un token de acceso al registro privado solo en el proyecto donde cree el flujo de imágenes, y configure ese flujo de imágenes con una política de referencia local. Conceda derechos para usar el flujo de imágenes a fin de atender a las cuentas de cada proyecto que use el flujo de imágenes.

La primera opción es similar a lo que haría para implementar una aplicación desde una imagen de contenedor en un registro privado. Niega algunas de las ventajas de usar flujos de imágenes, porque si la etiqueta de flujo de imágenes a la que hace referencia cambia para hacer referencia a otro servidor de registro, debe crear un nuevo secreto para el nuevo servidor de registro en todos los proyectos.

La segunda opción permite que los proyectos que hacen referencia al flujo de imágenes permanezcan aislados de los cambios en las etiquetas de flujo de imágenes que consumen. El trabajo adicional de asignar permisos a cuentas de servicio proviene del hecho de que las cuentas de servicio tienen derechos más restringidos que la cuenta de usuario que las crea.

En el ejemplo siguiente, se demuestra la segunda opción. Se crea un flujo de imágenes en el proyecto shared (compartido) y se usa ese flujo de imágenes para implementar una aplicación en el proyecto myapp.

```
[user@host ~]$ podman login -u myuser registry.example.com
[user@host ~]$ oc project shared
[user@host ~]$ oc create secret generic regtoken \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
[user@host ~]$ oc import-image myis --confirm \
> --reference-policy local \ ①
> --from registry.example.com/myorg/myimage
[user@host ~]$ oc policy add-role-to-group system:image-puller \ ②
> system:serviceaccounts:myapp \ ③
[user@host ~]$ oc project myapp
[user@host ~]$ oc new-app --as-deployment-config -i shared/myis
```

Hay tres detalles cruciales en el ejemplo anterior:

- ① La opción `--reference-policy local` del comando `oc import-image`. Configura el flujo de imágenes para almacenar en caché las capas de imagen del registro interno, por lo que los proyectos que hacen referencia al flujo de imágenes no necesitan un token de acceso al registro privado externo.

- ❷ El rol `system:image-puller` permite que una cuenta de servicio extraiga las capas de imagen que el flujo de imágenes almacena en caché en el registro interno.
- ❸ El grupo `system:serviceaccounts:myapp`. Este grupo incluye todas las cuentas de servicio del proyecto `myapp`. El comando `oc policy` puede hacer referencia a usuarios y grupos que aún no existen.

El comando `oc policy` no requiere privilegios de administrador de clúster; solo requiere privilegios de administrador de proyecto.

El flujo de imágenes del ejemplo anterior también puede proporcionar la imagen de compilador para el comando `oc new-app --as-deployment-config shared/myis-giturl`.



Referencias

Para obtener más información sobre los flujos de imágenes, las etiquetas de flujo de imágenes y los identificadores de imagen, consulte el capítulo *Análisis de contenedores, imágenes y flujos de imágenes* de la guía *Imágenes para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/images/index#understanding-images

► Ejercicio Guiado

Creación de un flujo de imagen

En este ejercicio, implementará una aplicación "hello, world" basada en Nginx, con un flujo de imágenes.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Publicar una imagen de un registro externo como un flujo de imágenes en OpenShift.
- Implementar una aplicación con el flujo de imágenes.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen de contenedor de aplicaciones "hello, world" (`redhattraining/hello-world-nginx`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos:

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

- 1. Inicie sesión en OpenShift y cree un proyecto para alojar el flujo de imágenes para la imagen de contenedor Nginx.

- 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 OpenShift con su cuenta de usuario de desarrollador.

```
[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 proyecto para alojar los flujos de imágenes que potencialmente se comparten entre varios proyectos:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-common
Now using project "youruser-common" on server
"https://api.cluster.domain.example.com:6443".
```

- 2. Cree un flujo de imágenes que apunte a la imagen Nginx del registro externo.

- 2.1. Compruebe que la imagen `redhattraining/hello-world-nginx` de Quay.io tenga una sola etiqueta denominada `latest`.

```
[student@workstation ~]$ skopeo inspect \
> docker://quay.io/redhattraining/hello-world-nginx
{
  "Name": "quay.io/redhattraining/hello-world-nginx",
  "Tag": "latest",
  "Digest": "sha256:4f4f...acc1",
  "RepoTags": [
    "latest"
  ],
  ...output omitted...
```

- 2.2. Cree la secuencia de imágenes `hello-world` que apunta a la imagen de contenedor `redhattraining/hello-world-nginx` de Quay.io:

```
[student@workstation ~]$ oc import-image hello-world --confirm \
> --from quay.io/redhattraining/hello-world-nginx
imagestream.image.openshift.io/hello-world imported
...output omitted...
Name:          hello-world
Namespace:     youruser-common
...output omitted...
Unique Images: 1
Tags:          1

latest
  tagged from quay.io/redhattraining/hello-world-nginx
...output omitted...
```

- 2.3. Verifique que se cree la etiqueta de flujo de imágenes `hello-world:latest`:

```
[student@workstation ~]$ oc get istag
NAME           IMAGE REF
hello-world:latest  quay.io/redhattraining/hello-world-nginx@sha256:4f4f...acc1
...  
...
```

- 2.4. Verifique que el flujo de imágenes y su etiqueta contengan metadatos acerca de la imagen de contenedor Nginx:

```
[student@workstation ~]$ oc describe is hello-world
Name:          hello-world
Namespace:     youruser-common
...output omitted...
```

```
Tags:          1

latest
tagged from quay.io/redhattraining/hello-world-nginx ①

* quay.io/redhattraining/hello-world-nginx@sha256:4f4f...acc1②
  2 minutes ago

...output omitted...
```

- ① La etiqueta de flujo de imágenes `hello-world:latest` hace referencia a una imagen de Quay.io.
- ② El asterisco (*) indica que la etiqueta de flujo de imágenes `latest` hace referencia solo a la imagen concreta con el identificador SHA-256 especificado.

► 3. Cree un proyecto nuevo e implemente una aplicación con el flujo de imágenes `hello-world` del proyecto `youruser-common`.

3.1. Cree un proyecto para alojar la aplicación de prueba:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-image-stream
Now using project "youruser-image-stream" on server
"https://api.cluster.domain.example.com:6443".
```

3.2. Implemente una aplicación con el flujo de imágenes:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name hello \
> -i ${RHT_OCP4_DEV_USER}-common/hello-world
--> Found image 44eaa13 (20 hours old) in image stream "youruser-common/hello-world" under tag "latest" for "youruser-common/hello-world"
...output omitted...
--> Creating resources ...
  imagestreamtag.image.openshift.io "hello:latest" created
  deploymentconfig.apps.openshift.io "hello" created
  service "hello" created
--> Success
...output omitted...
```

3.3. Espere hasta que el pod de aplicación esté listo y en ejecución:

```
[student@workstation ~]$ oc get pod
NAME           READY   STATUS    RESTARTS   AGE
hello-1-deploy 1/1     Completed  0          1m
hello-1-tzjwn  1/1     Running   0          1m
```

3.4. Cree una ruta para exponer la aplicación:

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

3.5. Obtenga el nombre de host de la ruta:

```
[student@workstation ~]$ oc get route  
NAME      HOST/PORT  
hello     hello-youruser-image-stream.apps.cluster.domain.example.com ...
```

3.6. Pruebe la aplicación con el comando `curl` y el nombre de host del paso anterior.

```
[student@workstation ~]$ curl \  
> http://hello-${RHT_OCP4_DEV_USER}-image-stream.${RHT_OCP4_WILDCARD_DOMAIN}  
...output omitted...  
    <h1>Hello, world from nginx!</h1>  
...output omitted...
```

► 4. Elimine los proyectos `youruser-commons` y `youruser-image-stream`.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-image-stream  
project.project.openshift.io "youruser-image-stream" deleted  
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-common  
project.project.openshift.io "youruser-image-common" deleted
```

Finalizar

En la máquina virtual `workstation`, ejecute el comando `lab image-stream finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab image-stream finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Publicación de imágenes de contenedor empresariales

Listado de verificación de rendimiento

En este trabajo de laboratorio, publicará una imagen de contenedor con formato OCI en un registro externo e implementará una aplicación a partir de esa imagen con un flujo de imágenes.



nota

El comando `grade` que se usa al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se declaran en la especificación del trabajo de laboratorio.

Resultados

Deberá ser capaz de crear una aplicación a partir de una imagen almacenada en un registro externo.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Los archivos que cumplen con OCI para la imagen de contenedor de muestra (`php-info`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

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

Requisitos

La imagen de la aplicación contiene una aplicación PHP que muestra el entorno del servidor web y la configuración del intérprete PHP. Implemente la aplicación de la siguiente manera:

- Aloje el flujo de imágenes para la imagen compilada fuera de OpenShift en un proyecto llamado `youruser-common`.

El nombre de la imagen de contenedor es `php-info`. Las capas de imagen con formato OCI y el manifiesto se encuentran en la carpeta `/home/student/D0288/labs/expose-image/php-info`. Envíe esa imagen a un repositorio de imágenes privado de su cuenta de Quay.io.

- Implemente la aplicación en un proyecto denominado `youruser-expose-image`.

Los recursos de la aplicación se denominan info. Acceda a la aplicación con el nombre de host predeterminado asignado por OpenShift.

- Use el archivo de configuración /usr/local/etc/ocp4.config para obtener datos de configuración del aula, como la URL de la API maestra del clúster OpenShift.

Pasos

1. Envíe la imagen de contenedor con formato OCI php-info a Quay.io.
2. Como usuario desarrollador, cree el proyecto *youruser-common* para alojar el flujo de imágenes que apunta a la imagen del registro externo. Cree un secreto con credenciales de inicio de sesión para Quay.io y cree el flujo de imágenes *php-info*.
Cree el flujo de imágenes con la opción `--reference-policy local` para que otros proyectos que usan ese flujo de imágenes también puedan usar el secreto almacenado en el proyecto *youruser-common*.
3. Cree un proyecto nuevo con el nombre *youruser-expose-image*. Luego cree una aplicación nueva mediante la implementación de la imagen de contenedor del flujo de imágenes.
Conceda el rol `system:image-puller` en el proyecto *youruser-common* a todas las cuentas de servicio del proyecto *youruser-expose-image*. Este rol permite que los pods de un proyecto usen flujos de imágenes de otro proyecto. El script `grant-puller-role.sh` de la carpeta `/home/student/D0288/labs/expose-image` contiene el comando `oc policy` que realiza esta operación.
4. Exponga y pruebe la aplicación. Verifique que la aplicación arroje la página de configuración del interpretador PHP.
5. Califique su trabajo.
Ejecute el siguiente comando en la máquina virtual `workstation` para verificar que se hayan realizado todas las tareas:

```
[student@workstation ~]$ lab expose-image grade
```

6. Elimine los proyectos de OpenShift y quite el repositorio de imágenes de Quay.io.

Finalizar

En la máquina virtual `workstation`, ejecute el comando `lab expose-image finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab expose-image finish
```

Esto concluye el trabajo de laboratorio.

► Solución

Publicación de imágenes de contenedor empresariales

Lista de verificación de rendimiento

En este trabajo de laboratorio, publicará una imagen de contenedor con formato OCI en un registro externo e implementará una aplicación a partir de esa imagen con un flujo de imágenes.



nota

El comando `grade` que se usa al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se declaran en la especificación del trabajo de laboratorio.

Resultados

Deberá ser capaz de crear una aplicación a partir de una imagen almacenada en un registro externo.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Los archivos que cumplen con OCI para la imagen de contenedor de muestra (`php-info`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

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

Requisitos

La imagen de la aplicación contiene una aplicación PHP que muestra el entorno del servidor web y la configuración del intérprete PHP. Implemente la aplicación de la siguiente manera:

- Aloje el flujo de imágenes para la imagen compilada fuera de OpenShift en un proyecto llamado `youruser-common`.

El nombre de la imagen de contenedor es `php-info`. Las capas de imagen con formato OCI y el manifiesto se encuentran en la carpeta `/home/student/D0288/labs/expose-image/php-info`. Envíe esa imagen a un repositorio de imágenes privado de su cuenta de Quay.io.

- Implemente la aplicación en un proyecto denominado `youruser-expose-image`.

Los recursos de la aplicación se denominan info. Acceda a la aplicación con el nombre de host predeterminado asignado por OpenShift.

- Use el archivo de configuración /usr/local/etc/ocp4.config para obtener datos de configuración del aula, como la URL de la API maestra del clúster OpenShift.

Pasos

1. Envíe la imagen de contenedor con formato OCI php-info a Quay.io.

- 1.1. Verifique que la carpeta php-info contenga una imagen de contenedor con formato OCI:

```
[student@workstation ~]$ ls ~/D0288/labs/expose-image/php-info
blobs    index.json    oci-layout
```

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

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

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

- 1.3. Use el comando podman para iniciar sesión en Quay.io.

```
[student@workstation ~]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 1.4. Use el comando skopeo para enviar la imagen de contenedor con formato OCI a Quay.io. Puede copiar o ejecutar el comando skopeo del script push-image.sh en la carpeta /home/student/D0288/labs/expose-image.

```
[student@workstation ~]$ skopeo copy \
> oci:/home/student/D0288/labs/expose-image/php-info \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/php-info
...output omitted...
Writing manifest to image destination
Storing signatures
```

- 1.5. Inspeccione la imagen en el registro externo con Skopeo para verificar que esté etiquetada como latest.

```
[student@workstation ~]$ skopeo inspect \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/php-info
{
  "Name": "quay.io/yourquayuser/php-info",
  "Tag": "latest",
  ...output omitted...
```

2. Como usuario desarrollador, cree el proyecto `youruser-common` para alojar el flujo de imágenes que apunta a la imagen del registro externo. Cree un secreto con credenciales de inicio de sesión para Quay.io y cree el flujo de imágenes `php-info`.

Cree el flujo de imágenes con la opción `--reference-policy local` para que otros proyectos que usan ese flujo de imágenes también puedan usar el secreto almacenado en el proyecto `youruser-common`.

- 2.1. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 2.2. Cree un proyecto para alojar el flujo de imágenes.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-common
Now using project "youruser-common" on server
"https://api.cluster.domain.example.com:6443".
```

- 2.3. Cree un secreto con el token de acceso de la API de registro de contenedores almacenado por Podman.

Puede copiar o ejecutar el comando `oc create secret` del script `create-secret.sh` en la carpeta `/home/student/D0288/labs/expose-image`.

```
[student@workstation ~]$ oc create secret generic quayio \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
secret/quayio created
```

- 2.4. Importe la imagen de contenedor con el comando `oc import-image`:

```
[student@workstation ~]$ oc import-image php-info --confirm \
> --reference-policy local \
> --from quay.io/${RHT_OCP4_QUAY_USER}/php-info
imagestream.image.openshift.io/php-info imported
...output omitted...
latest
tagged from quay.io/youruser/php-info
will use insecure HTTPS or HTTP connections

* quay.io/youruser/php-info@sha256:4366...f937
  Less than a second ago
...output omitted...
```

- 2.5. Verifique que se haya creado una etiqueta de flujo de imágenes que contenga metadatos acerca de la imagen de contenedor `php-info`:

```
[student@workstation ~]$ oc get istag
NAME          IMAGE REF    ...
php-info:latest  image-registry.openshift-image-registry.svc:5000/youruser-
common/php-info@sha256:4366...f937  ...
```

3. Cree un proyecto nuevo con el nombre *youruser-expose-image*. Luego cree una aplicación nueva mediante la implementación de la imagen de contenedor del flujo de imágenes.

Conceda el rol *system:image-puller* en el proyecto *youruser-common* a todas las cuentas de servicio del proyecto *youruser-expose-image*. Este rol permite que los pods de un proyecto usen flujos de imágenes de otro proyecto. El script *grant-puller-role.sh* de la carpeta */home/student/D0288/labs/expose-image* contiene el comando *oc policy* que realiza esta operación.

- 3.1. Cree un proyecto para alojar la aplicación:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-expose-image
```

- 3.2. Conceda a las cuentas de servicio del nuevo proyecto *youruser-expose-image* acceso a los flujos de imágenes del proyecto *youruser-common*. Puede copiar o ejecutar el siguiente comando *oc policy* del script *grant-puller-role.sh* en la carpeta */home/student/D0288/labs/expose-image*.

```
[student@workstation ~]$ oc policy add-role-to-group \
> -n ${RHT_OCP4_DEV_USER}-common system:image-puller \
> system:serviceaccounts:${RHT_OCP4_DEV_USER}-expose-image
clusterrole.rbac.authorization.k8s.io/system:image-puller added:
"system:serviceaccounts:youruser-expose-image"
```

- 3.3. Implemente la aplicación de prueba del flujo de imágenes en el proyecto *common*:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name info \
> -i ${RHT_OCP4_DEV_USER}-common/php-info
...output omitted...
--> Creating resources ...
imagestreamtag.image.openshift.io "info:latest" created
deploymentconfig.apps.openshift.io "info" created
service "info" created
--> Success
...output omitted...
```

- 3.4. Espere a que el pod de la aplicación esté listo y en ejecución:

```
[student@workstation ~]$ oc get pod
NAME        READY   STATUS    RESTARTS   AGE
info-1-deploy  1/1     Running   0          22s
info-1-80rkd   1/1     Running   0          22s
```

4. Exponga y pruebe la aplicación. Verifique que la aplicación arroje la página de configuración del interpretador PHP.

- 4.1. Exponga la aplicación info:

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

- 4.2. Obtenga el nombre de host que OpenShift asigna a la ruta:

```
[student@workstation ~]$ oc get route info
NAME      HOST/PORT
info      info-youruser-expose-image.apps.cluster.domain.example.com
```

- 4.3. Pruebe la aplicación con el comando `curl` y la ruta del paso anterior. Como alternativa, puede usar un navegador web.

```
[student@workstation ~]$ curl \
> http://info-${RHT_OCP4_DEV_USER}-expose-image.${RHT_OCP4_WILDCARD_DOMAIN}
...output omitted...
<title>phpinfo()</title><meta name="ROBOTS" content="NOINDEX, NOFOLLOW, NOARCHIVE" /></head>
...output omitted...
<tr><td class="e">Server API </td><td class="v">FPM/FastCGI </td></tr>
...output omitted...
<tr><td class="e">PHP API </td><td class="v">20170718 </td></tr>
...output omitted...
```

5. Califique su trabajo.

Ejecute el siguiente comando en la máquina virtual `workstation` para verificar que se hayan realizado todas las tareas:

```
[student@workstation ~]$ lab expose-image grade
```

6. Elimine los proyectos de OpenShift y quite el repositorio de imágenes de Quay.io.

- 6.1. Elimine el proyecto `youruser-expose-image`:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-expose-image
project.project.openshift.io "youruser-expose-image" deleted
```

- 6.2. Elimine el proyecto `youruser-common`:

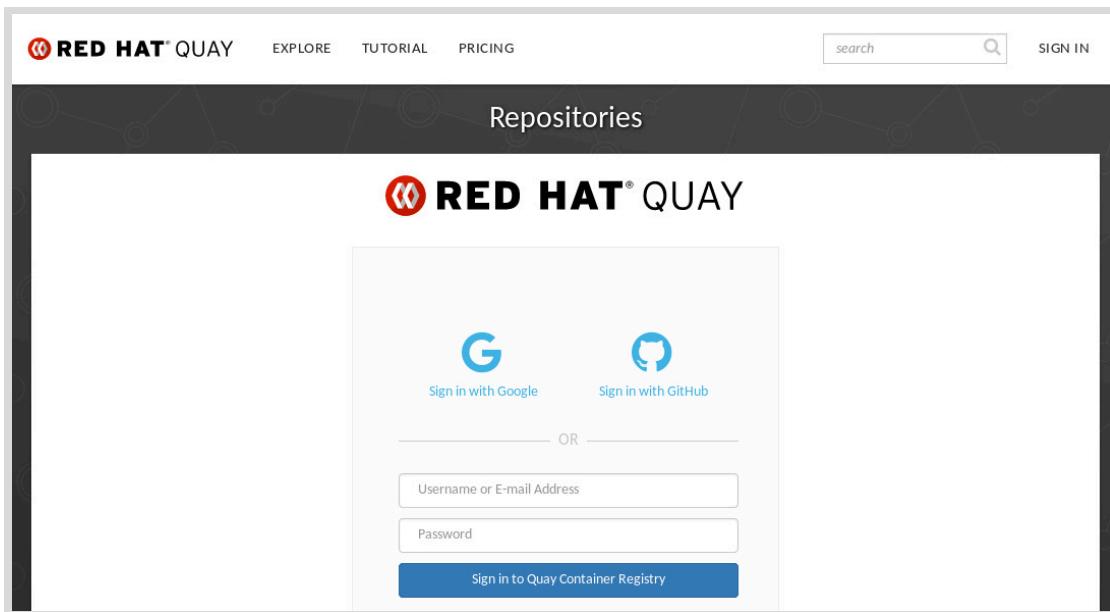
```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-common
project.project.openshift.io "youruser-common" deleted
```

- 6.3. Elimine la imagen de contenedor del registro externo:

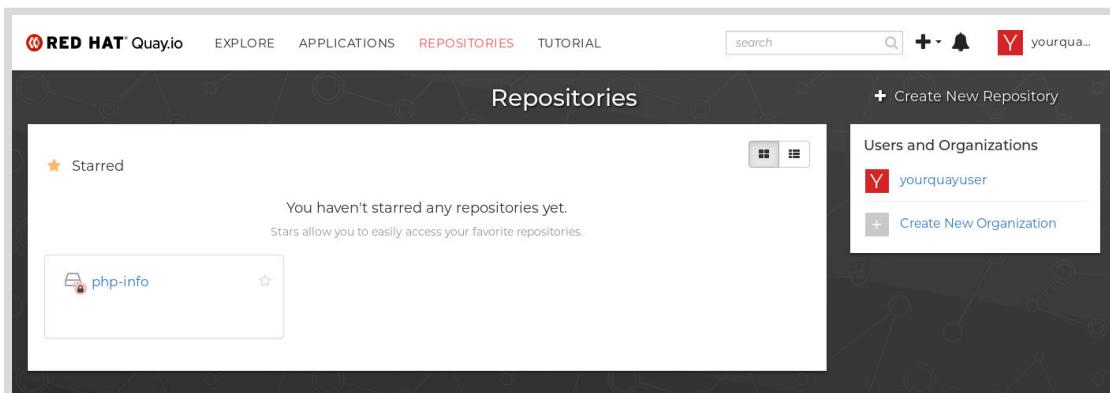
```
[student@workstation ~]$ skopeo delete \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/php-info:latest
```

- 6.4. Inicie sesión en Quay.io con su cuenta personal gratuita.

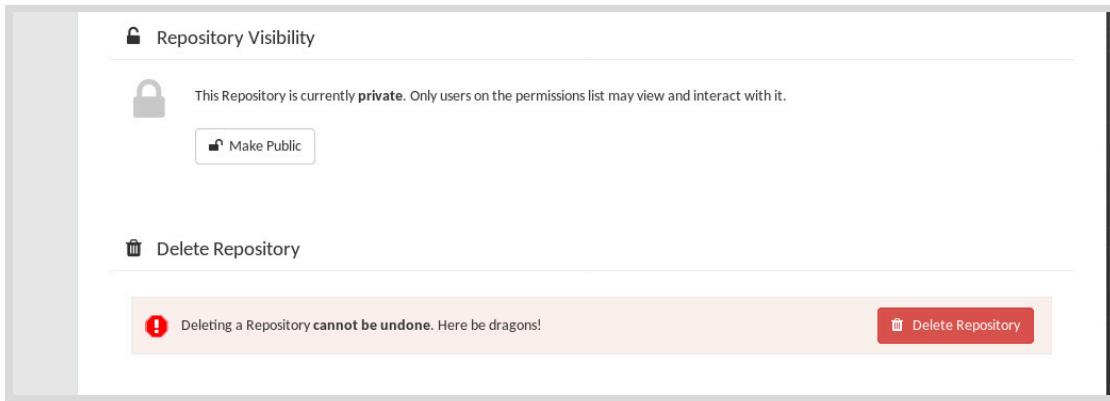
Desplácese hasta <http://quay.io> y haga clic en **Sign In** (Iniciar sesión) para proporcionar sus credenciales de usuario. Haga clic en **Sign in to Quay Container Registry** (Iniciar sesión en Quay Container Registry) para iniciar sesión en Quay.io.



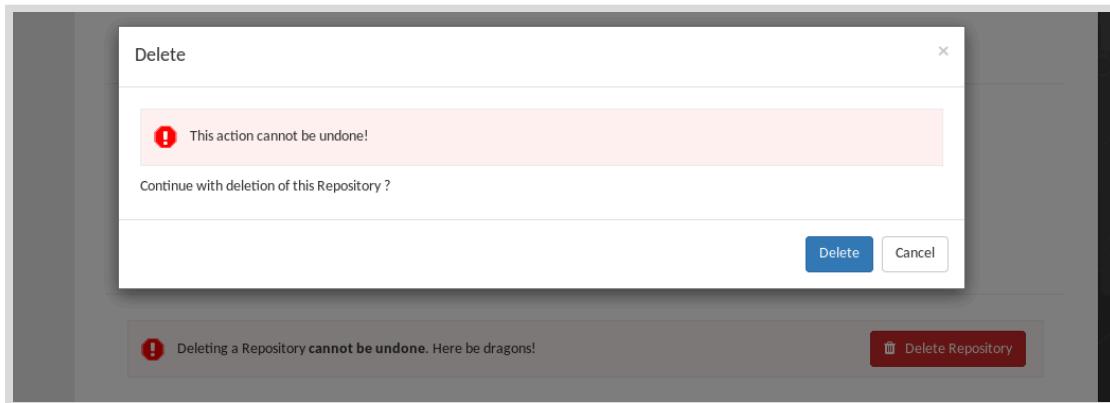
- 6.5. En el menú principal de Quay.io, haga clic en **Repositories** (Repositorios) y busque **php-info**. Haga clic en **php-info** para abrir la página **Repository Activity** (Actividad del repositorio).



- 6.6. En la página **Repository Activity** (Actividad del repositorio) del repositorio **php-info**, desplácese hacia abajo y haga clic en el ícono de engranaje para mostrar la pestaña **Settings** (Configuración). Desplácese hacia abajo y haga clic en **Delete Repository** (Eliminar repositorio).



- 6.7. En el cuadro de diálogo **Delete** (Eliminar), haga clic en **Delete** para confirmar que desea eliminar el repositorio `php - info`. Después de unos momentos, volverá a la página **Repositories** (Repositorios). Ahora puede cerrar sesión en Quay.io.



Finalizar

En la máquina virtual `workstation`, ejecute el comando `lab expose-image finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab expose-image finish
```

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Los desarrolladores de OpenShift interactúan con muchos tipos de servidores de registro que pueden o no requerir autenticación, incluido el registro interno de OpenShift.
- OpenShift requiere que los desarrolladores creen secretos para almacenar tokens de acceso a registros privados y externos.
- Las herramientas de contenedor de Linux (Skopeo, Podman y Buildah) pueden acceder al registro interno de OpenShift, como cualquier otro servidor de registro, con el token de acceso de usuario de OpenShift, ya sea como un registro seguro o inseguro.
- Los flujos de imágenes de OpenShift y los recursos de etiquetas de flujo de imágenes proporcionan referencias estables a las imágenes de contenedor y también aíslan a los desarrolladores de las direcciones del servidor de registro.

Compilación de aplicaciones

Meta

Describir el proceso de compilación de OpenShift, desencadenar y administrar compilaciones.

Objetivos

- Describir el proceso de compilación de OpenShift.
- Administrar las compilaciones de la aplicación mediante el recurso BuildConfig y los comandos de CLI.
- Desencadenar el proceso de compilación con métodos soportados.
- Procesar lógicas posteriores a la compilación con un hook de compilación post-commit (posterior a la confirmación).

Secciones

- Descripción del proceso de compilación de OpenShift (y cuestionario)
- Administración de las compilaciones de la aplicación (y ejercicio guiado)
- Desencadenamiento de compilaciones (y ejercicio guiado)
- Implementación de hooks de compilación post-commit (posteriores a la confirmación) (y ejercicio guiado)

Trabajo de laboratorio

Compilación de aplicaciones

Descripción del proceso de compilación de OpenShift

Objetivos

Después de completar esta sección, deberá ser capaz de describir el proceso de compilación de OpenShift.

El proceso de compilación

El proceso de compilación de Red Hat OpenShift Container Platform transforma el código fuente o los archivos binarios y otros parámetros de entrada en imágenes de contenedor que están disponibles para su implementación en la plataforma. Para compilar una imagen de contenedor, OpenShift requiere un recurso `BuildConfig`. La configuración de este recurso incluye una estrategia y una o más fuentes de entrada, como definiciones Git, binarias o en línea.

Estrategias de compilación

Las siguientes son las estrategias de compilación disponibles en OpenShift:

- Origen
- Pipeline
- Docker
- Custom

Cada estrategia requiere una imagen de contenedor.

Origen

La estrategia `Source` crea una nueva imagen de contenedor en función del código fuente de la aplicación o binarios de aplicación. OpenShift clona el código fuente de la aplicación o copia los binarios de aplicación en una imagen de compilador compatible y arma una nueva imagen de contenedor lista para implementarse en la plataforma.

Esta estrategia simplifica la manera en que los desarrolladores compilan imágenes de contenedor debido a que funciona con las herramientas con las que están familiarizados, en lugar de usar comandos de SO de nivel bajo, como `yum` en `Dockerfiles`.

OpenShift basa la estrategia `Source` en el proceso fuente a imagen (S2I).

Pipeline

La estrategia `JenkinsPipeline` crea una nueva imagen de contenedor mediante el complemento (plug-in) de tubería Jenkins. Aunque Jenkins compila las imágenes de contenedor, OpenShift puede iniciar, monitorear y administrar la compilación.

El recurso `BuildConfig` hace referencia a un `Jenkinsfile` que contiene los flujos de trabajo de tubería. Puede definir el `Jenkinsfile` directamente en la configuración de compilación o extraerlo de un repositorio Git externo.

OpenShift inicia un nuevo servidor Jenkins para ejecutar la tubería en la primera compilación mediante la estrategia Pipeline. Las configuraciones de compilación de tuberías posteriores comparten este servidor Jenkins.

Docker

La estrategia Docker usa el comando `buildah` para compilar una nueva imagen de contenedor con un archivo `Dockerfile`. La estrategia Docker puede recuperar el `Dockerfile` y los artefactos para compilar la imagen de contenedor desde un repositorio Git, o puede usar un `Dockerfile` provisto en línea en la configuración de compilación como fuente de compilación.

La compilación de Docker se ejecuta como un pod dentro del clúster OpenShift. Los desarrolladores no necesitan tener herramientas de Docker en su estación de trabajo.



nota

Las compilaciones de Docker requieren privilegios elevados y es posible que el administrador de clústeres de OpenShift deniegue a algunos o todos los usuarios el derecho a iniciar compilaciones de Docker.

Custom

La estrategia Custom especifica una imagen de generador responsable del proceso de compilación. Esta estrategia permite a los desarrolladores personalizar el proceso de compilación. Consulte la sección de referencias de esta unidad para obtener más información sobre cómo crear una imagen de generador personalizada.

Compilación de fuentes de entrada

La compilación de fuentes de entrada proporciona contenido de origen para las compilaciones. OpenShift admite los siguientes seis tipos de fuentes de entrada, enumerados en orden de prioridad:

- **Dockerfile**: Especifica el Dockerfile en línea para compilar una imagen.
- **Image**: Puede proporcionar archivos adicionales al proceso de compilación a partir de imágenes.
- **Git**: OpenShift clona el código fuente de la aplicación de entrada desde un repositorio Git. Es posible configurar la ubicación predeterminada dentro del repositorio donde la compilación busca el código fuente de la aplicación.
- **Binary**: Permite la transmisión de contenido binario de un sistema de archivos local al compilador.
- **Input secrets**: Puede usar secretos de entrada para permitir la creación de credenciales para la compilación que no están disponibles en la imagen de la aplicación final.
- **External artifacts**: Permite copiar archivos binarios al proceso de compilación.

Puede combinar varias entradas en una sola compilación. Sin embargo, como el Dockerfile en línea tiene prioridad, invalida cualquier otro Dockerfile proporcionado por otra entrada. Además, la entrada binaria y los repositorios Git son entradas mutuamente excluyentes.

**nota**

Aunque OpenShift ofrece muchas estrategias y fuentes de entrada, los escenarios más frecuentes son usar las estrategias Source o Docker, con un repositorio Git como la fuente de entrada.

Recurso BuildConfig

La configuración de compilación define la manera en que se realiza el proceso de compilación. El recurso **BuildConfig** define una única configuración de compilación y un conjunto de desencadenadores para cuando OpenShift debe crear una nueva compilación.

OpenShift genera **BuildConfig** mediante el comando `oc new-app`. También se puede generar mediante el botón **Add to Project** (Aregar al proyecto) desde la consola web o creando una aplicación desde una plantilla.

El siguiente ejemplo compila una aplicación PHP mediante la estrategia Source y una fuente de entrada Git:

```
{
  "kind": "BuildConfig",
  "apiVersion": "v1",
  "metadata": {
    "name": "php-example", ①
    ...
  },
  "spec": {
    "triggers": [ ②
      {
        "type": "GitHub",
        "github": {
          "secret": "gukAWHzq1On4AJlMjvjs" ③
        }
      },
      ...
    ],
    "runPolicy": "Serial", ④
    "source": { ⑤
      "type": "Git",
      "git": {
        "uri": "http://services.lab.example.com/php-helloworld"
      }
    },
    "strategy": { ⑥
      "type": "Source",
      "sourceStrategy": {
        "from": {
          "kind": "ImageStreamTag",
          "namespace": "openshift",
          "name": "php:7.0"
        }
      }
    },
    "output": { ⑦
    }
  }
}
```

```

    "to": {
      "kind": "ImageStreamTag",
      "name": "php-example:latest"
    }
  },
  ...
},
...
}

```

- ➊ Define un nuevo `BuildConfig` denominado `php-example`.
- ➋ Define los desencadenadores que inician las nuevas compilaciones.
- ➌ La cadena de autorización para el webhook, generada aleatoriamente por OpenShift. Las aplicaciones externas envían esta cadena como parte de la URL del webhook para desencadenar nuevas compilaciones.
- ➍ El atributo `runPolicy` define si una compilación se puede iniciar simultáneamente. El valor `Serial` representa que no se puede compilar simultáneamente.
- ➎ El atributo `source` es responsable de definir la fuente de entrada de la compilación. En este ejemplo, usa un repositorio Git.
- ➏ Define la estrategia de compilación para compilar la imagen de contenedor final. En este ejemplo, usa la estrategia `Source`.
- ➐ El atributo `output` define dónde se envía la nueva imagen de contenedor después de una compilación exitosa.



Referencias

Para obtener más información sobre las imágenes de compilación personalizadas y fuentes de entrada de compilación, consulte el capítulo *Creación de entradas de compilación* de la guía *Compilaciones para Red Hat OpenShift Container Platform 4.5* en

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/builds/creating-build-inputs

Las compilaciones de fuente versus las compilaciones binarias S2I se explican con más detalle en

<https://developers.redhat.com/blog/2018/09/26/source-versus-binary-s2i-workflows-with-red-hat-openshift-application-runtimes/>

► Cuestionario

El proceso de compilación de OpenShift

Elija las respuestas correctas para las siguientes preguntas:

► 1. ¿Cuáles tres de las siguientes opciones son estrategias válidas para compilar una imagen de contenedor? (Elija tres opciones).

- a. Docker
- b. Git
- c. Pipeline
- d. Custom

► 2. ¿Cuáles dos de las siguientes opciones son tipos de fuentes de entrada válidos para compilar una imagen de contenedor? (Elija dos opciones).

- a. Git
- b. SVN
- c. Sistema de archivos
- d. Dockerfile

► 3. Dado `BuildConfig` en la siguiente muestra, ¿cuáles de las siguientes son las tres afirmaciones verdaderas? (Elija tres opciones).

```
{
  "kind": "BuildConfig",
  "apiVersion": "v1",
  "metadata": {
    "name": "php-example",
  },
  "spec": {
    ...
    "runPolicy": "Serial",
    "source": {
      "type": "Git",
      "git": {
        "uri": "http://services.lab.example.com/php-helloworld"
      }
    },
    "strategy": {
      "type": "Source",
      "sourceStrategy": {
        "from": {
          "kind": "ImageStreamTag",
          "namespace": "openshift",
          "name": "php:7.0"
        }
      }
    },
    "output": {
      "to": {
        "kind": "ImageStreamTag",
        "name": "php-example:latest"
      }
    },
    ...
  },
}
```

- Se pueden ejecutar múltiples compilaciones al mismo tiempo.
- Después de una compilación exitosa, la imagen de contenedor `php-example:latest` está disponible para la implementación.
- La compilación usa una fuente de entrada Git para crear la imagen de contenedor final.
- `BuildConfig` usa la estrategia Docker.
- `BuildConfig` usa la estrategia Source.

- 4. Para la estrategia Docker, ¿cuál de las siguientes es la fuente de entrada que se declara en una configuración de compilación generada por el comando `oc new-app`?
- a. Dockerfile
 - b. Binary
 - c. Git
 - d. Ninguna de las anteriores
- 5. Un desarrollador desea compilar una aplicación PHP mediante la estrategia Source. ¿Cuál de las siguientes opciones configura la configuración de compilación para usar la imagen del compilador S2I para una aplicación PHP?
- a. Use el atributo `from` para la estrategia Source.
 - b. Use el atributo `from` de la fuente de entrada `Image`.
 - c. Use el atributo `from` de la fuente de entrada `Dockerfile`.
 - d. No se requiere la configuración de la imagen de compilador S2I. Durante la compilación, el proceso de fuente a imagen reconocerá el lenguaje fuente y seleccionará una imagen del compilador PHP automáticamente.
 - e. Ninguna de estas opciones.

► Solución

El proceso de compilación de OpenShift

Elija las respuestas correctas para las siguientes preguntas:

- ▶ 1. **¿Cuáles tres de las siguientes opciones son estrategias válidas para compilar una imagen de contenedor? (Elija tres opciones).**
 - a. Docker
 - b. Git
 - c. Pipeline
 - d. Custom

- ▶ 2. **¿Cuáles dos de las siguientes opciones son tipos de fuentes de entrada válidos para compilar una imagen de contenedor? (Elija dos opciones).**
 - a. Git
 - b. SVN
 - c. Sistema de archivos
 - d. Dockerfile

► 3. Dado `BuildConfig` en la siguiente muestra, ¿cuáles de las siguientes son las tres afirmaciones verdaderas? (Elija tres opciones).

```
{  
    "kind": "BuildConfig",  
    "apiVersion": "v1",  
    "metadata": {  
        "name": "php-example",  
    },  
    "spec": {  
        ...  
        "runPolicy": "Serial",  
        "source": {  
            "type": "Git",  
            "git": {  
                "uri": "http://services.lab.example.com/php-helloworld"  
            }  
        },  
        "strategy": {  
            "type": "Source",  
            "sourceStrategy": {  
                "from": {  
                    "kind": "ImageStreamTag",  
                    "namespace": "openshift",  
                    "name": "php:7.0"  
                }  
            }  
        },  
        "output": {  
            "to": {  
                "kind": "ImageStreamTag",  
                "name": "php-example:latest"  
            }  
        },  
        ...  
    },  
}
```

- a. Se pueden ejecutar múltiples compilaciones al mismo tiempo.
- b. Después de una compilación exitosa, la imagen de contenedor `php-example:latest` está disponible para la implementación.
- c. La compilación usa una fuente de entrada Git para crear la imagen de contenedor final.
- d. `BuildConfig` usa la estrategia Docker.
- e. `BuildConfig` usa la estrategia Source.

► **4. Para la estrategia Docker, ¿cuál de las siguientes es la fuente de entrada que se declara en una configuración de compilación generada por el comando oc new-app?**

- a. Dockerfile
- b. Binary
- c. Git
- d. Ninguna de las anteriores

► **5. Un desarrollador desea compilar una aplicación PHP mediante la estrategia Source.**

¿Cuál de las siguientes opciones configura la configuración de compilación para usar la imagen del compilador S2I para una aplicación PHP?

- a. Use el atributo `from` para la estrategia Source.
- b. Use el atributo `from` de la fuente de entrada Image.
- c. Use el atributo `from` de la fuente de entrada Dockerfile.
- d. No se requiere la configuración de la imagen de compilador S2I. Durante la compilación, el proceso de fuente a imagen reconocerá el lenguaje fuente y seleccionará una imagen del compilador PHP automáticamente.
- e. Ninguna de estas opciones.

Administración de compilaciones de una aplicación

Objetivos

Después de completar esta sección, deberá ser capaz de administrar las compilaciones de aplicaciones mediante el recurso de configuración de compilación y los comandos de la CLI.

Crear una configuración de compilación

Red Hat OpenShift Container Platform administra compilaciones a través del recurso de configuración de compilación. Hay dos maneras de crear una configuración de compilación:

Usar el comando `oc new-app`

Este comando puede crear una configuración de compilación en función de los argumentos especificados. Por ejemplo, si se define un repositorio Git, se crea una configuración de compilación mediante la estrategia fuente. También, si una plantilla es el argumento y la plantilla tiene una configuración de compilación definida, crea una configuración de compilación basada en los parámetros de plantilla.

Usar una configuración de compilación personalizada

Cree una configuración de compilación personalizada con sintaxis YAML o JSON e impórtela a OpenShift con el comando `oc create -f`.

Administrar compilaciones de una aplicación mediante la CLI

OpenShift proporciona varias opciones que permiten que los desarrolladores administren las compilaciones de aplicaciones y las configuraciones de compilación mediante la CLI. Estos comandos se usan para administrar el ciclo de vida de una aplicación, especialmente durante el desarrollo. Tenga en cuenta que la configuración de compilación es una fase independiente en comparación con la implementación. Una compilación correcta en OpenShift da como resultado una nueva imagen de contenedor, que desencadena una nueva implementación.

`oc start-build`

Inicia una nueva compilación manualmente. El nombre del recurso de configuración de compilación es el único argumento requerido para iniciar una nueva compilación.

```
[user@host ~]$ oc start-build name
```

Una compilación correcta crea una nueva imagen de contenedor en la salida ImageStreamTag. Si una configuración de implementación define un desencadenador en ImageStreamTag, se inicia el proceso de implementación.

`oc cancel-build`

Cancela una compilación. Si una compilación se inicia con una versión incorrecta de la aplicación, por ejemplo, y la aplicación no puede implementarse, es posible que deba cancelarse la compilación antes de que falle.

Solo se pueden cancelar compilaciones que estén en estado de ejecución o pendientes. Cancelar una compilación significa que el pod de compilación finaliza, de modo que no

hay una nueva imagen de contenedor para enviar. Por lo tanto, la implementación no se desencadena.

Proporcione el nombre del recurso de configuración de compilación para cancelar una compilación:

```
[user@host ~]$ oc cancel-build name
```

oc delete

Elimina una configuración de compilación. Por lo general, una configuración de compilación se elimina cuando necesita importar una nuevo desde un archivo.

```
[user@host ~]$ oc delete bc/name
```

Este comando también elimina una compilación (no una configuración de compilación) para recuperar el espacio que usa esta. Una configuración de compilación puede tener varias compilaciones. Proporcione el nombre de la compilación para eliminar una compilación:

```
[user@host ~]$ oc delete build/name-1
```

oc describe

Describe detalles acerca de un recurso de configuración de compilación y acerca de compilaciones asociadas, lo que le proporciona información, como etiquetas, estrategia, webhook, etc.

```
[user@host ~]$ oc describe bc name
```

Describa una compilación proporcionando el nombre de la compilación:

```
[user@host ~]$ oc describe build name-1
```

oc logs

Muestra los registros de compilación. Puede verificar si su aplicación se está compilando correctamente. Este comando también muestra registros de una compilación finalizada. No es posible verificar los registros de compilaciones eliminadas o reducidas. Hay dos maneras de mostrar un registro de compilación:

- Mostrar los registros de compilación de la compilación más reciente.

```
[user@host ~]$ oc logs -f bc/name
```

La opción -f sigue el registro, hasta que finaliza el comando con Ctrl+C.

- Mostrar los registros de compilación de una compilación específica:

```
[user@host ~]$ oc logs build/name-1
```

Eliminación de compilaciones

De forma predeterminada, las compilaciones que se han completado se conservan indefinidamente. Puede limitar el número de compilaciones anteriores que se conservan mediante

el atributo `successfulBuildsHistoryLimit` y el atributo `failedBuildsHistoryLimit`, como se muestra en el siguiente fragmento de código YAML de una configuración de compilación:

```
apiVersion: "v1"
kind: "BuildConfig"
metadata:
  name: "sample-build"
spec:
  successfulBuildsHistoryLimit: 2
  failedBuildsHistoryLimit: 2
```

Las compilaciones con errores incluyen los estados `fallida`, `cancelada` o `error`. Las compilaciones se ordenan por hora de creación, y las compilaciones más antiguas se eliminan primero.



nota

Los administradores de OpenShift pueden eliminar manualmente las compilaciones mediante el comando de eliminación de objetos `oc adm`.

Detalles del registro

OpenShift proporciona dos mecanismos diferentes para configurar los detalles del registro de compilación. El primero es global, y un administrador puede definir la configuración de los detalles de registro de la compilación para todo el clúster. El segundo solo afecta los detalles del registro de compilación de las compilaciones a partir de una configuración de compilación específica. Esto significa que los desarrolladores mantienen la capacidad de invalidar la configuración global para una configuración de compilación más específica cuando requieren un nivel diferente de detalle del registro.

Edite el recurso de configuración de compilación y agregue la variable de entorno `BUILD_LOGLEVEL` como parte de la estrategia fuente o estrategia Docker para configurar un nivel de registro específico:

```
[user@host ~]$ oc set env bc/name BUILD_LOGLEVEL="4"
```

El valor debe ser un número entre cero y cinco. Cero es el valor predeterminado y muestra menos de cinco registros. Cuando aumenta el número, el detalle de los mensajes de registro es mayor y los registros contienen más detalles.

Eliminación de compilaciones

"Eliminar" una compilación significa suprimir una compilación completada o fallida. OpenShift puede hacerlo automáticamente según la configuración, o puede eliminar manualmente una compilación con los comandos `oc delete` o `oc adm prune`.

Los administradores puede eliminar compilaciones y liberar espacio en disco para evitar la acumulación de versiones anteriores en el almacén de datos `etcd`. Hay algunas opciones de configuración para eliminar compilaciones, como eliminación de huérfanos, eliminación basada en la cantidad de compilaciones correctas, eliminación basada en la cantidad de compilaciones fallidas, y más.

Es posible que a veces necesite pedirles a los administradores que cambien la configuración de eliminación para que los registros de compilación se conserven durante más tiempo que el

predeterminado y pueda solucionar problemas de compilación. Estos temas están fuera del ámbito del curso actual.



Referencias

Para obtener más información sobre cómo gestionar las compilaciones de la aplicación, consulte el capítulo *Cómo realizar una compilación básica* de la guía *Compilaciones para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/builds/basic-build-operations

► Ejercicio Guiado

Administración de compilaciones de una aplicación

En este ejercicio, administrará compilaciones de una aplicación con OpenShift mediante la estrategia fuente con una fuente de entrada Git.

Resultados

Usted deberá ser capaz de usar la CLI para manejar las compilaciones en OpenShift.

Antes De Comenzar

Para realizar este ejercicio, necesita asegurarse de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador OpenJDK 1.8 S2I y el flujo de imágenes (`redhat-openjdk18-openshift:1.5`).
- La aplicación de muestra del repositorio Git (`java-serverhost`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos requeridos para completar este ejercicio:

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

- 1. Inspeccione el código fuente de Java para obtener la aplicación de muestra.

- 1.1. Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación `master` del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 1.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b manage-builds
Switched to a new branch 'manage-builds'
[student@workstation D0288-apps]$ git push -u origin manage-builds
...output omitted...
 * [new branch]      manage-builds -> manage-builds
Branch manage-builds set up to track remote branch manage-builds from origin.
```

- 1.3. Revise el código fuente de Java para la aplicación, en la carpeta `java-serverhost`.

capítulo 4 | Compilación de aplicaciones

Abra el archivo `ServerHostEndPoint.java` en la carpeta `/home/student/D0288-apps/java-serverhost/src/main/java/com/redhat/training/example/javaserverhost/rest:`

```
package com.redhat.training.example.javaserverhost.rest;

import javax.ws.rs.Path;
import javax.ws.rs.core.Response;
import javax.ws.rs.GET;
import javax.ws.rs.Produces;
import java.net.InetAddress;

@Path("/")
public class ServerHostEndPoint {

    @GET
    @Produces("text/plain")
    public Response doGet() {
        String host = "";
        try {
            host = InetAddress.getLocalHost().getHostName();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        String msg = "I am running on server "+host+" Version 1.0 \n";
        return Response.ok(msg).build();
    }
}
```

La aplicación implementa un servicio REST simple que devuelve el nombre de host del servidor en el que se ejecuta.

► 2. Cree un proyecto nuevo.

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 D0288-apps]$ source /usr/local/etc/ocp4.config
```

2.2. Inicie sesión en OpenShift con su usuario de desarrollador:

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
```

2.3. Cree un nuevo proyecto para alojar la aplicación:

```
[student@workstation D0288-apps]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-manage-builds
```

► 3. Cree una nueva aplicación.

- 3.1. Cree una nueva aplicación a partir de las fuentes de Git. Use la bifurcación que creó en el paso anterior. Asígnele el nombre `jhost` a la aplicación y use la opción `--build-env` del comando `oc new-app` para definir una variable de entorno de compilación con la ubicación del repositorio maven.

El comando completo puede copiarse o ejecutarse directamente desde el script `oc-new-app.sh` en la carpeta `/home/student/D0288/labs/manage-builds`:

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config --name jhost \
> --build-env MAVEN_MIRROR_URL=http:// ${RHT_OCP4_NEXUS_SERVER}/repository/java \
> -i redhat-openjdk18-openshift:1.5 \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#manage-builds \
> --context-dir java-serverhost
...output omitted...
imagestream.image.openshift.io "jhost" created
buildconfig.build.openshift.io "jhost" created
deploymentconfig.apps.openshift.io "jhost" created
service "jhost" created
...output omitted...
```

Observe que no hay espacio antes ni después del signo igual (=) después de `MAVEN_MIRROR_URL`. El par key=value completo para la variable de entorno de compilación es demasiado largo para el ancho del papel.

- 3.2. Verifique los registros de compilación de la compilación `jhost` con el comando `oc logs`:

```
[student@workstation D0288-apps]$ oc logs -f bc/jhost
...output omitted...
Writing manifest to image destination
Storing signatures
...output omitted...
Push successful
```

- 3.3. Espere a que la aplicación esté lista y en ejecución:

```
[student@workstation D0288-apps]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
jhost-1-build  0/1     Completed  0          32m
jhost-1-deploy 0/1     Completed  0          30m
jhost-1-10mbp  1/1     Running   0          30m
```

- 3.4. Exponga la aplicación a acceso externo:

```
[student@workstation D0288-apps]$ oc expose svc/jhost
route.route.openshift.io/jhost exposed
```

- 3.5. Obtenga el nombre de host generado por OpenShift para la nueva ruta:

```
[student@workstation D0288-apps]$ oc get route
NAME      HOST/PORT   ...
jhost    jhost-youruser-manage-builds.apps.cluster.domain.example.com ...
```

3.6. Pruebe la aplicación:

```
[student@workstation D0288-apps]$ curl \
> http://jhost-${RHT_OCP4_DEV_USER}-manage-builds.${RHT_OCP4_WILDCARD_DOMAIN}
I am running on server jhost-1-10mbp Version 1.0
```

**nota**

Aun si el contenedor de la aplicación está listo, la aplicación puede tardar algunos segundos en responder. Si el comando anterior devuelve una página HTML con el texto **Application is not available** (La aplicación no está disponible), espere unos segundos e inténtelo de nuevo.

▶ 4. Enumere configuraciones de compilación y compilaciones.

- 4.1. Enumere todas las configuraciones de compilación en el proyecto al ejecutar el comando `oc get bc`:

```
[student@workstation D0288-apps]$ oc get bc
NAME      TYPE      FROM          LATEST
jhost     Source    Git@manage-builds  1
```

El comando `oc new-app` creó un recurso e configuración de compilación denominado `jhost` mediante la estrategia `Source` con una fuente de entrada `Git`.

- 4.2. La configuración de compilación creada por el comando `oc new-app` inició una compilación. Enumere todas las compilaciones disponibles en el proyecto:

```
[student@workstation D0288-apps]$ oc get builds
NAME      TYPE      FROM          STATUS      STARTED           DURATION
jhost-1   Source    Git@cb73a3d  Complete    18 minutes ago  2m4s
```

▶ 5. Actualice la aplicación con la versión 2.0.

- 5.1. Edite el archivo `ServerHostEndPoint.java` en la carpeta `/home/student/D0288-apps/java-serverhost/src/main/java/com/redhat/training/example/javaserverhost/rest` y actualice a la versión 2.0:

```
String msg = "I am running on server "+host+" Version 2.0 \n";
```

- 5.2. Confirme los cambios al servidor Git.

```
[student@workstation D0288-apps]$ cd java-serverhost
[student@workstation java-serverhost]$ git commit -a -m 'Update the version'
...output omitted...
1 file changed, 1 insertion(+), 1 deletion(-)
```

- 5.3. Inicie una nueva compilación con el comando `oc start-build`:

```
[student@workstation java-serverhost]$ oc start-build bc/jhost
build.build.openshift.io/jhost-2 started
```

- 5.4. La aplicación que se está compilando todavía es la anterior, sin los cambios que confirmó en su repositorio Git local. Necesita enviarlos antes de iniciar la compilación de OpenShift. Cancele la compilación para evitar una nueva implementación mediante las fuentes anteriores.

```
[student@workstation java-serverhost]$ oc cancel-build bc/jhost
build.build.openshift.io/jhost-2 marked for cancellation, waiting to be cancelled
build.build.openshift.io/jhost-2 cancelled
```

- 5.5. Compruebe que se haya cancelado la compilación:

```
[student@workstation java-serverhost]$ oc get builds
NAME      TYPE      FROM      STATUS      ...
jhost-1   Source    Git@cb73a3d Complete   ...
jhost-2   Source    Git@cb73a3d Cancelled (CancelledBuild) ...
```

- 5.6. Inserte el código fuente actualizado en el servidor Git:

```
[student@workstation java-serverhost]$ git push
...output omitted...
2aa4b3a..f70977b manage-builds -> manage-builds
[student@workstation java-serverhost]$ cd ~
```

- 5.7. Inicie una nueva compilación para implementar la versión actualizada de la aplicación:

```
[student@workstation ~]$ oc start-build bc/jhost
build.build.openshift.io/jhost-3 started
```

- 5.8. Enumere todas las compilaciones:

```
[student@workstation ~]$ oc get builds
NAME      TYPE      FROM      STATUS      ...
jhost-1   Source    Git@cb73a3d Complete   ...
jhost-2   Source    Git@cb73a3d Cancelled (CancelledBuild) ...
jhost-3   Source    Git        Running    ...
```

Observe que hay tres compilaciones disponibles. La primera fue creada por el comando `oc new-app`, la segunda la canceló usted y la tercera tiene la aplicación actualizada.

- 5.9. Siga los registros de compilación de la aplicación:

```
[student@workstation ~]$ oc logs -f build/jhost-3
...output omitted...
Push successful
```

- 5.10. Espere a que la aplicación esté lista y en ejecución:

```
[student@workstation ~]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
jhost-1-build  0/1     Completed  0          53m
jhost-1-deploy 0/1     Completed  0          51m
jhost-2-29zh5  1/1     Running   0          4m40s
jhost-2-deploy 0/1     Completed  0          4m49s
jhost-3-build  0/1     Completed  0          6m21s
```

5.11. Pruebe la aplicación actualizada:

```
[student@workstation ~]$ curl \
> http://jhost-${RHT_OCP4_DEV_USER}-manage-builds.${RHT_OCP4_WILDCARD_DOMAIN}
I am running on server jhost-2-10mbp Version 2.0
```

► 6. Limpie y elimine el proyecto:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-manage-builds
project.project.openshift.io "youruser-manage-builds" deleted
```

Finalizar

En workstation, ejecute el script lab manage-builds finish para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

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

Esto concluye el ejercicio guiado.

Desencadenamiento de compilaciones

Objetivos

Después de completar esta sección, deberá ser capaz de desencadenar el proceso de compilación con los métodos soportados.

Definición de desencadenadores de compilación

En OpenShift puede definir desencadenadores de compilación para permitir que la plataforma inicie nuevas compilaciones automáticamente en función de determinados eventos. Puede usar estos desencadenadores de compilación para mantener los contenedores de la aplicación actualizados con las nuevas imágenes de contenedor o cambios de código que afecten a la aplicación. OpenShift define dos tipos de desencadenadores de compilación:

Desencadenadores de cambio de imagen

Un desencadenador de cambio de imagen vuelve a compilar una imagen de contenedor de aplicaciones para incorporar cambios realizados por su imagen principal.

Desencadenadores de webhooks

Los desencadenadores de webhooks de OpenShift son extremos (endpoints) API HTTP que inician una nueva compilación. Use un webhook para integrar OpenShift con el sistema de control de versiones (VCS), como Github o BitBucket, para desencadenar nuevas compilaciones en los cambios de código.

Los desencadenadores de cambio de imagen permiten que un desarrollador no tenga que buscar cambios en la imagen principal de una aplicación. Los desencadenadores de cambio de imagen se pueden activar automáticamente mediante el registro interno de OpenShift cuando detecta una nueva imagen. Si la imagen procede de un registro externo, debe ejecutar periódicamente el comando `oc import-image` para comprobar si la imagen del contenedor ha cambiado en el servidor de registro para mantenerse al día.

El comando `oc new-app` crea desencadenadores de cambio de imagen automáticamente para las aplicaciones mediante estrategias de compilación fuente o Docker:

- Para la estrategia fuente, la imagen principal es la imagen del compilador S2I para el lenguaje de programación de la aplicación.
- Para la estrategia Docker, la imagen principal es la imagen referenciada por la instrucción `FROM` en el Dockerfile de la aplicación.

Para ver los desencadenadores asociados a una configuración de compilación, use el comando `oc describe bc`, como se muestra en el ejemplo siguiente:

```
[user@host ~]$ oc describe bc/name
```

Para agregar un desencadenador de cambio de imagen a una configuración de compilación, use el comando `oc set triggers`:

```
[user@host ~]$ oc set triggers bc/name --from-image=project/image:tag
```

Una sola configuración de compilación no puede incluir varios desencadenadores de cambio de imagen.

Para eliminar un desencadenador de cambio de imagen de una configuración de compilación, use el comando `oc set triggers` con la opción `--remove`:

```
[user@host ~]$ oc set triggers bc/name --from-image=project/image:tag --remove
```

Use el comando `oc set triggers --help` para ver las opciones usadas para agregar y eliminar un desencadenador de cambio de configuración.

Inicio de nuevas compilaciones con desencadenadores de webhook

Los desencadenadores de webhook de OpenShift son extremos (endpoints) API HTTP que inician nuevas compilaciones. Use un webhook para integrar OpenShift con un sistema de control de versiones (VCS), como Git, de modo que los cambios en el código fuente de la aplicación desencadenen una nueva compilación en OpenShift con el último código.

El software no tiene que ser un VCS para usar estos extremos (endpoints) API, pero las compilaciones de OpenShift solo pueden buscar código fuente de un servidor Git.

Red Hat OpenShift Container Platform proporciona tipos de webhook especializados que admiten extremos (endpoints) API compatibles con los siguientes servicios de VCS:

- GitLab
- GitHub
- Bitbucket

Red Hat OpenShift Container Platform también proporciona un tipo de webhook genérico que adopta una carga útil definida por OpenShift. Un webhook genérico puede ser usado por cualquier software para iniciar una compilación de OpenShift. Consulte las referencias de documentación del producto al final de esta sección para obtener la sintaxis de la carga útil del webhook genérico y las solicitudes de API HTTP para cada tipo de webhook.

Para todos los webhooks, debe definir un secreto con una clave denominada `WebHookSecretKey` y el valor es el valor que se va a proporcionar al invocar el webhook. A continuación, la definición de webhook debe hacer referencia al secreto. El secreto garantiza la unicidad de la dirección URL, lo que impide que otros desencadenen la compilación. El valor de la clave se comparará con el secreto proporcionado durante la invocación del webhook. Cada vez que se crea un desencadenador o que OpenShift crea uno automáticamente, también se crea un secreto de forma predeterminada.

El comando `oc new-app` crea un webhook genérico y un webhook Git. Para agregar otros tipos de webhook a una configuración de compilación, use el comando `oc set triggers`. Por ejemplo, para agregar un webhook de GitLab a una configuración de compilación:

```
[user@host ~]$ oc set triggers bc/name --from-gitlab
```

Si la configuración de compilación ya incluye un webhook de GitLab, el comando anterior restablece el secreto de autenticación incrustado en la URL. Debe actualizar sus proyectos de GitLab para usar la nueva URL del webhook.

Para eliminar un webhook existente de una configuración de compilación, use el comando `oc set triggers` con la opción `--remove`. Por ejemplo, para eliminar un webhook de GitLab de una configuración de compilación:

```
[user@host ~]$ oc set triggers bc/name --from-gitlab --remove
```

El comando `oc set triggers` también admite las opciones `--from-github` y `--from-bitbucket` para crear los desencadenadores específicos de cada plataforma VCS.

Para recuperar la URL de un enlace web y el secreto, use el comando `oc describe` y busque el tipo específico de enlace web que necesita.



Referencias

Puede obtener más información sobre los desencadenadores de compilaciones en el capítulo *Desencadenamiento y modificación de compilaciones* de la guía *Compilaciones para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/builds/triggering-builds-build-hooks

► Ejercicio Guiado

Desencadenamiento de compilaciones

En este ejercicio, desencadenará una nueva compilación de una aplicación en OpenShift para incorporar actualizaciones realizadas en la imagen del compilador S2I de la aplicación.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear una aplicación que se base en una imagen del compilador S2I.
- Enviar una nueva versión de la imagen del compilador S2I.
- Actualizar los metadatos que se encuentran dentro de un flujo de imágenes para actuar sobre la actualización de la imagen.
- Verificar que la aplicación se vuelve a compilar para usar la nueva imagen del compilador S2I.

Antes De Comenzar

Para realizar este ejercicio, necesita asegurarse de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La versión original de la imagen del compilador S2I PHP (`php-70-rhel7-original.tar.gz`)
- La nueva versión de la imagen del compilador S2I PHP (`php-70-rhel7-newer.tar.gz`)
- La aplicación de muestra del repositorio Git (`trigger-builds`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab trigger-builds start
```

Pasos

► 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 OpenShift con su usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
```

- 1.3. Cree un nuevo proyecto para la aplicación:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-trigger-builds
```

- 2. Agregue un flujo de imágenes al proyecto que se usará con la nueva aplicación.

- 2.1. Inicie sesión en su cuenta personal de Quay.io con Podman.

```
[student@workstation ~]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 2.2. Envíe la imagen original del compilador PHP 7.0 a su registro público Quay.io.

Los comandos que necesita para insertar la imagen están disponibles en el script `push-original.sh` del directorio `/home/student/D0288/labs/trigger-builds`. Puede copiar y pegar los comandos en el script, o puede ejecutar el script.

```
[student@workstation ~]$ cd /home/student/D0288/labs/trigger-builds
[student@workstation trigger-builds]$ skopeo copy \
> docker-archive:php-70-rhel7-original.tar.gz \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/php-70-rhel7:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

- 2.3. El registro de Quay.io tiene, de forma predeterminada, imágenes privadas, por lo que tendrá que agregar un secreto a una cuenta de servicio del compilador para tener acceso a ella.

Cree un secreto con el token de acceso de la API de registro de contenedores almacenado por Podman.

```
[student@workstation trigger-builds]$ oc create secret generic quay-registry \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
secret/quay-registry created
```

Agregue el secreto de registro de Quay.io a la cuenta de servicio del compilador.

```
[student@workstation trigger-builds]$ oc secrets link builder quay-registry
```

- 2.4. Actualice el flujo de imágenes php para buscar los metadatos para la nueva imagen de contenedor. El registro externo usa el paquete `docker-distribution` y no notifica a OpenShift acerca de los cambios en las imágenes.

```
[student@workstation trigger-builds]$ oc import-image php \
> --from quay.io/${RHT_OCP4_QUAY_USER}/php-70-rhel7 --confirm
imagestream.image.openshift.io/php-70-rhel7 imported

Name:          php
...output omitted...
latest
tagged from quay.io/youruser/php-70-rhel7

* quay.io/youruser/php-70-rhel7@sha256:c919...8cb2
  Less than a second ago
...output omitted...
```

► 3. Cree una nueva aplicación.

- 3.1. Cree una nueva aplicación a partir de las fuentes de Git. Asígnale el nombre `trigger` a la aplicación.

El comando completo puede copiarse o ejecutarse directamente desde el script `oc-new-app.sh` en la carpeta `/home/student/D0288/labs/trigger-builds`:

```
[student@workstation trigger-builds]$ oc new-app --as-deployment-config \
> --name trigger \
> php~http://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
> --context-dir trigger-builds
--> Found image ... "youruser-trigger-builds/php" ... for "php"

...output omitted...

--> Success
Build scheduled, use 'oc logs -f bc/trigger' to track its progress.
...output omitted...
```

- 3.2. Espere a que finalice la compilación.

```
[student@workstation trigger-builds]$ oc logs -f bc/trigger
...output omitted...
Push successful
```

- 3.3. Espere a que la aplicación esté lista y en ejecución:

```
[student@workstation trigger-builds]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
trigger-1-build  0/1     Completed   0          10m
trigger-1-deploy 0/1     Completed   0          9m10s
trigger-1-q6bln  1/1     Running    0          9m2s
```

- 3.4. Verifique que se define un desencadenador de cambio de imagen como parte de la configuración de compilación:

```
[student@workstation trigger-builds]$ oc describe bc/trigger | grep Triggered
Triggered by: Config, ImageChange
```

El último desencadenador, el desencadenador **ImageChange**, inicia una nueva compilación si el flujo de imágenes detecta que su imagen base ha cambiado.

► 4. Actualice el flujo de imágenes para iniciar una nueva compilación.

- 4.1. Cargue la nueva versión de la imagen del compilador PHP S2I en el registro Quay.io

Los comandos que necesita para insertar la nueva imagen están disponibles en el script `push-newer.sh` del directorio `/home/student/D0288/labs/trigger-builds`. Puede copiar y pegar los comandos en el script, o puede ejecutar el script.

```
[student@workstation trigger-builds]$ skopeo copy \
> docker-archive:php-70-rhel7-newer.tar.gz \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/php-70-rhel7:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

- 4.2. Actualice el flujo de imágenes `php` para buscar los metadatos para la nueva imagen de contenedor.

```
[student@workstation trigger-builds]$ oc import-image php
imagestream.image.openshift.io/php-70-rhel7 imported

Name:          php
...output omitted...
latest
tagged from quay.io/youruser/php-70-rhel7

* quay.io/youruser/php-70-rhel7@sha256:3f5e...6ab7
  Less than a second ago
quay.io/youruser/php-70-rhel7@sha256:c919...8cb2
  2 minutes ago
...output omitted...
```

► 5. Revise la nueva compilación.

- 5.1. La actualización del flujo de imágenes desencadenó una nueva compilación. Enumere todas las compilaciones para verificar que se haya iniciado una segunda compilación:

```
[student@workstation trigger-builds]$ oc get builds
NAME      TYPE      FROM      STATUS      STARTED      DURATION
trigger-1  Source   Git@da010df  Complete   13 minutes ago  58s
trigger-2  Source   Git@da010df  Complete   28 seconds ago  15s
```

- 5.2. Confirme que la compilación `trigger-2` haya sido iniciada por una actualización en el flujo de imágenes:

```
[student@workstation trigger-builds]$ oc describe build trigger-2 | grep cause
Build trigger cause: Image change
```

► 6. Finalice.

- 6.1. Pase al directorio principal.

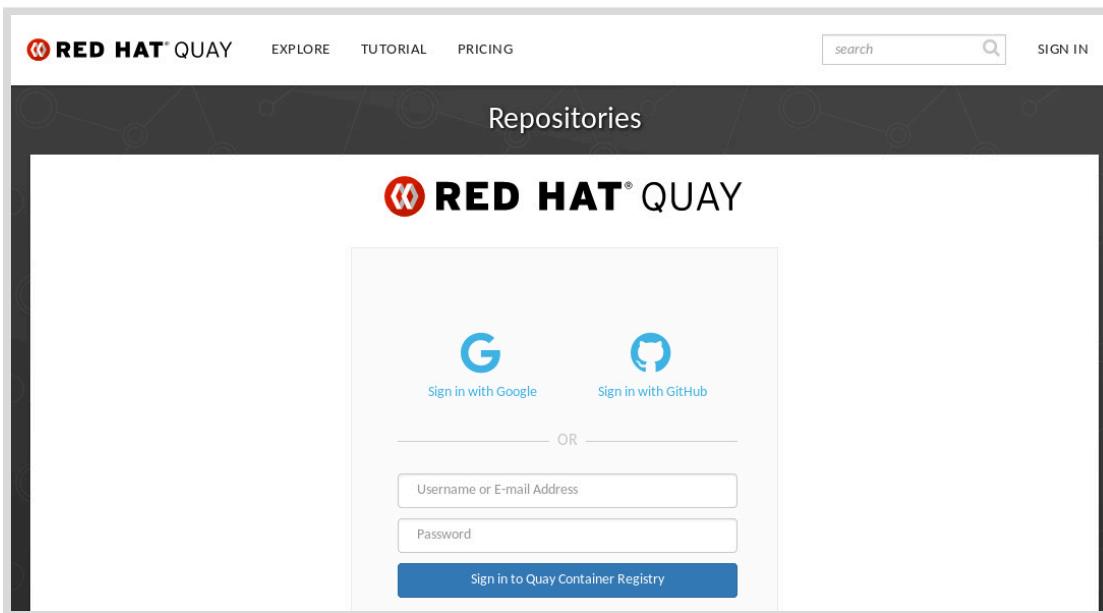
```
[student@workstation trigger-builds]$ cd ~
[student@workstation ~]$
```

- 6.2. Elimine la imagen de contenedor del registro externo:

```
[student@workstation ~]$ skopeo delete \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/php-70-rhel7
```

- 6.3. Inicie sesión en Quay.io con su cuenta personal gratuita.

Visite <http://quay.io> and y haga clic en **Sign In** (Iniciar sesión) para proporcionar sus credenciales de usuario. Haga clic en **Sign in to Quay Container Registry** (Iniciar sesión en Quay Container Registry) para iniciar sesión en Quay.io.



- 6.4. En el menú de nivel superior de Quay.io, haga clic en **Repositories** (Repositorios) y busque **php-70-rhel7**. El ícono de candado que está al lado indica que es un repositorio privado que requiere autenticación para extracciones e inserciones. Haga clic en **php-70-rhel7** para ingresar en la página **Repository Activity** (Actividad del repositorio).

The screenshot shows the Red Hat Quay interface with the 'Repositories' page. At the top, there are navigation links for EXPLORE, APPLICATIONS, REPOSITORIES (which is highlighted in red), and TUTORIAL. There are also icons for a plus sign, a bell, and a user profile. On the right side, there's a sidebar titled 'Users and Organizations' with a section for 'yourquayuser' and a link to 'Create New Organization'.

- 6.5. En la página **Repository Activity** (Actividad del repositorio) del repositorio **php-70-rhel7**, desplácese hacia abajo hasta ver el ícono de engranaje y haga clic para ingresar en la pestaña **Settings** (Configuración). Desplácese hacia abajo hasta que encuentre el botón **Delete Repository** (Eliminar repositorio) y haga clic en él.

This screenshot shows the 'Repository Activity' page for the 'php-70-rhel7' repository. It includes sections for 'Repository Visibility' (private) and 'Delete Repository'. A warning message at the bottom states: 'Deleting a Repository cannot be undone. Here be dragons!'. A red 'Delete Repository' button is visible.

- 6.6. En la ventana emergente **Delete** (Eliminar), haga clic en **Delete** (Eliminar) para confirmar que desea eliminar el repositorio **php-70-rhel7**. Después de unos momentos, volverá a la página **Repositories** (Repositorios). Ahora puede cerrar sesión en Quay.io.

This screenshot shows a confirmation dialog box titled 'Delete'. It contains a warning message: 'This action cannot be undone!', a question 'Continue with deletion of this Repository?', and two buttons: 'Delete' (blue) and 'Cancel' (white). Below the dialog, a message bar repeats the warning: 'Deleting a Repository cannot be undone. Here be dragons!'. A red 'Delete Repository' button is also visible.

Finalizar

En `workstation`, ejecute el script `lab trigger-builds finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab trigger-builds finish
```

Esto concluye el ejercicio guiado.

Implementación de hooks de compilación post-commit (posteriores a la confirmación)

Objetivos

Tras finalizar esta sección, deberá ser capaz de procesar la lógica posterior a la compilación con un hook de compilación post-commit (posterior a la confirmación).

Descripción de hooks de compilación post-commit (posteriores a la confirmación)

OpenShift proporciona una funcionalidad de hooks de compilación post-commit (posteriores a la confirmación) para realizar tareas de validación durante las compilaciones. El hook de compilación post-commit (posterior a la confirmación) ejecuta comandos en un contenedor temporal antes de enviar la nueva imagen de contenedor generada por la compilación al registro. El hook inicia un contenedor temporal a través de la imagen de contenedor creada por la compilación.

Según el código de salida de la ejecución del comando en el contenedor temporal, OpenShift enviará o no la imagen al registro. Si el comando devuelve un código de salida distinto de cero, indicando un error, OpenShift no envía la imagen y marca la compilación como fallida. Si el comando se ejecuta correctamente, OpenShift envía la imagen al registro.

Puede comprobar que una compilación ha fallado debido a un hook post-commit (posterior a la confirmación) examinando los registros de compilación mediante el comando `oc logs`:

```
[user@host ~]$ oc logs bc/name
...
Running post commit hook ...
...
error: build error: container "openshift_s2i-build_hook-2_post-commit_post-
commit_45ec0816" returned non-zero exit code: 35
```

Revisión de casos de uso para hooks de compilación posteriores a la confirmación

Un escenario típico para usar un hook de compilación post-commit (posterior a la confirmación) es ejecutar algunas pruebas en la aplicación. De este modo, antes de que OpenShift envíe la imagen al registro e inicie una nueva implementación, las pruebas pueden comprobar si la aplicación está funcionando correctamente. Si se produce un error en la prueba, se produce un error en la compilación y no continúa con una implementación.

Hay otros casos de uso comunes en los que puede ser útil aprovechar un hook de compilación post-commit (posterior a la confirmación). Por ejemplo:

- Integrar una compilación con una aplicación externa a través de API HTTP.
- Validar un requisito no funcional, como el rendimiento de la aplicación, la seguridad, la facilidad de uso o la compatibilidad.
- Enviar un correo electrónico al equipo del desarrollador informándole de la nueva compilación.

Configuración de un hook de compilación post-commit (posterior a la confirmación)

Hay dos tipos de hooks de compilación post-commit (posteriores a la confirmación) que puede configurar:

Comando

Un comando se ejecuta a través de la llamada del sistema exec. Cree un hook de compilación post-commit (posterior a la confirmación) con la opción `--command` como se muestra en el siguiente comando:

```
[user@host ~]$ oc set build-hook bc/name --post-commit \
> --command -- bundle exec rake test --verbose
```



nota

El espacio posterior al argumento `--` no es un error. El argumento `--` separa la línea de comandos de OpenShift que configura un hook post-commit (posterior a la confirmación) de un comando ejecutado por el hook.

Script de shell

Ejecuta un hook de compilación con el comando `/bin/sh -ic`. Esto es más práctico, ya que tiene todas las características proporcionadas por una shell, como expansión de argumento, redirección, etc. Solo funciona si la imagen base tiene la shell sh. Cree un enlace de compilación post-commit (posterior a la confirmación) del script de shell con `--script` como se muestra en el siguiente comando:

```
[user@host ~]$ oc set build-hook bc/name --post-commit \
> --script="curl http://api.com/user/${USER}"
```



Referencias

Para obtener más información sobre los hooks de compilación post-commit (posteriores a la confirmación), consulte el capítulo *Desencadenamiento y modificación de compilaciones* de la guía *Compilaciones para Red Hat OpenShift Container Platform 4.5* en

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/builds/triggering-builds-build-hooks

► Ejercicio Guiado

Implementación de hooks de compilación post-commit (posteriores a la confirmación)

En este ejercicio, definirá un hook de compilación post-commit (posterior a la confirmación) para integrar una compilación con una aplicación externa.

Resultados

Deberá ser capaz de agregar un hook de compilación post-commit (posterior a la confirmación) a un recurso de configuración de compilación. El hook de compilación post-commit (posterior a la confirmación) envía una solicitud de API HTTP a la aplicación `builds-for-managers`.

Antes De Comenzar

La aplicación `builds-for-managers` es usada por administradores para realizar un seguimiento de las compilaciones de la aplicación realizadas por un equipo de desarrolladores. En la aplicación se muestra información, como el proyecto, la URL de Git y el desarrollador que inició la compilación. Necesita integrar sus compilaciones con la aplicación para que los administradores estén al tanto de su trabajo.

Para realizar este ejercicio, necesita asegurarse de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I PHP (`php-73-rhel7:7.3`).
- La imagen de contenedor `builds-for-managers` (`quay.io/redhattraining/builds-for-managers`)
- El repositorio Git de la aplicación (`post-commit`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos, cree el proyecto `post-commit`, inicie la aplicación `builds-for-managers` y descargue los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab post-commit 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 D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 1.3. Asegúrese de que está en el proyecto `youruser-post-commit`:

```
[student@workstation ~]$ oc project ${RHT_OCP4_DEV_USER}-post-commit
Already on project "youruser-post-commit" on server "https://...".
```

Dado que el comando `lab post-commit start` crea este proyecto, ya debe estar en este proyecto. Si no es así, la salida puede diferir de la anterior.

- 1.4. Compruebe que `builds-for-managers` se está ejecutando en el proyecto `youruser-post-commit`:

```
[student@workstation ~]$ oc status
In project youruser-post-commit on server https://api.cluster.domain....
```

```
http://builds-for-managers... to pod port 8080-tcp (svc/builds-for-managers)
dc/builds-for-managers deploys istag/builds-for-managers:latest
  deployment #1 deployed 5 minutes ago - 1 pod
...output omitted...
```

► 2. Cree una nueva aplicación.

- 2.1. Cree una nueva aplicación a partir de las fuentes de Git. Asignele el nombre `hook` a la aplicación y anteponga el flujo de imágenes `php:7.3` a la URL del repositorio Git a través de una tilde (~).

El comando completo puede copiarse o ejecutarse directamente desde el script `oc-new-app.sh` en la carpeta `/home/student/D0288/labs/post-commit`:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name hook \
> php:7.3~http://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
> --context-dir post-commit
```

- 2.2. Espere a que finalice la compilación:

```
[student@workstation ~]$ oc logs -f bc/hook
...output omitted...
Push successful
```

- 2.3. Espere a que la aplicación esté lista y en ejecución:

```
[student@workstation ~]$ oc get pods
NAME                  READY   STATUS    RESTARTS   AGE
builds-for-managers-1-deploy  0/1     Completed   0          7m
builds-for-managers-1-w1s8f    1/1     Running    0          8m
hook-1-build              0/1     Completed   0          1m
hook-1-deploy              0/1     Completed   0          3m
hook-1-lmn12                1/1     Running    0          11s
```

Actualmente, hay dos aplicaciones diferentes en ejecución. La primera es la aplicación `builds-for-managers` usada por los administradores; la segunda es su aplicación PHP.

► 3. Integre la compilación con la aplicación PHP `builds-for-managers`.

- 3.1. Inspeccione el script `create-hook.sh` que se proporciona en la carpeta `/home/student/D0288/labs/post-commit`. Crea un hook de compilación que integra las compilaciones de su aplicación PHP con la aplicación `builds-for-managers` con el comando `curl`.

```
[student@workstation ~]$ cat ~/D0288/labs/post-commit/create-hook.sh
...output omitted...
oc set build-hook bc/hook --post-commit --command -- \
  bash -c "curl -s -S -i -X POST http://builds-for-managers-
${RHT_OCP4_DEV_USER}-post-commit.${RHT_OCP4_WILDCARD_DOMAIN}/api/builds
-f -d 'developer=\${DEVELOPER}&git=\${OPENSHIFT_BUILD_SOURCE}&project=\
\${OPENSHIFT_BUILD_NAMESPACE}'"
```

El comando `curl` envía tres variables de entorno a la aplicación `builds-for-managers`:

- `DEVELOPER`: Contiene el nombre del desarrollador.
- `OPENSHIFT_BUILD_SOURCE`: Contiene la URL de Git del proyecto.
- `OPENSHIFT_BUILD_NAMESPACE`: Contiene el nombre del proyecto.

- 3.2. Ejecute el script `create-hook.sh`:

```
[student@workstation ~]$ ~/D0288/labs/post-commit/create-hook.sh
buildconfig.build.openshift.io/hook hooks updated
```

- 3.3. Verifique que se haya creado el hook de compilación `post-commit` (posterior a la confirmación):

```
[student@workstation ~]$ oc describe bc/hook | grep Post
Post Commit Hook: ["bash", "-c", "\"curl -s -S -i -X POST http://builds-...
```

- 3.4. Inicie una nueva compilación con el comando `oc start-build` con la opción `-F` habilitada para mostrar los registros y verificar el código de respuesta de la API HTTP. Verá un código de estado HTTP 400 devuelto por el comando `curl` ejecutado por el hook `post-commit` (posterior a la confirmación):

```
[student@workstation ~]$ oc start-build bc/hook -F
build.build.openshift.io/hook-2 started
...output omitted...
STEP 11: RUN bash -c "curl -s -S -i -X POST http://builds-for-managers-...
curl: (22) The requested URL returned error: 400 Bad Request
subprocess exited with status 22
subprocess exited with status 22
error: build error: error building at STEP "RUN bash -c "curl -s -S -i ..."
```

La aplicación `builds-for-managers` rechazó la solicitud de la API HTTP porque no se definió la variable del entorno `DEVELOPER`.

- 3.5. Enumere las compilaciones y verifique que una compilación falló debido a un error en el hook post-commit (posterior a la confirmación):

```
[student@workstation ~]$ oc get builds
NAME      TYPE      FROM      STATUS    ...output omitted...
hook-1    Source    Git@c2166cc Complete
hook-2    Source    Git@c2166cc Failed (GenericBuildFailed)
```

► 4. Corrija el problema de la variable de entorno ausente.

- 4.1. Cree la variable del entorno de compilación `DEVELOPER` usando su nombre con el comando `oc set env`:

```
[student@workstation ~]$ oc set env bc/hook DEVELOPER="Your Name"
buildconfig.build.openshift.io/hook updated
```

- 4.2. Verifique que la variable del entorno `DEVELOPER` esté disponible en la configuración de compilación hook:

```
[student@workstation ~]$ oc set env bc/hook --list
# buildconfigs hook
DEVELOPER=Your Name
```

- 4.3. Inicie una nueva compilación y consulte los registros para verificar el código de respuesta de la API HTTP. Verá un código de estado HTTP 200:

```
[student@workstation ~]$ oc start-build bc/hook -F
build.build.openshift.io/hook-3 started
...output omitted...
STEP 11: RUN bash -c "curl -s -S -i -X POST http://builds-for-managers-...
HTTP/1.1 200 OK
...output omitted...
Push successful
```

El código de estado HTTP 200 indica que la aplicación `builds-for-managers` aceptó la solicitud de API HTTP.

- 4.4. Obtenga la ruta expuesta `builds-for-managers`:

```
[student@workstation ~]$ oc get route/builds-for-managers \
> -o jsonpath='{.spec.host}{"\n"}'
builds-for-managers-youruser-post-commit.apps.cluster.domain.example.com
```

- 4.5. Abra un navegador web para acceder a `http://builds-for-managers-youruser-post-commit.apps.cluster.domain.example.com`. En la página, se muestran todas las compilaciones y el desarrollador que inició cada una.

Builds for Managers

| Date | Developer | Project | Git Project |
|---------------------|-----------|----------------------|---|
| 2019-07-10 07:08:43 | Your Name | youruser-post-commit | http://github.com/youruser/DO288-apps |

- ▶ 5. Limpieza: Elimine el proyecto `youruser-post-commit` en OpenShift.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-post-commit
```

Finalizar

En `workstation`, ejecute el script `lab post-commit finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab post-commit finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Compilación de aplicaciones

Listado de verificación de rendimiento

En este trabajo de laboratorio, usará OpenShift para administrar las compilaciones de aplicaciones, solucionar problemas de una aplicación y desencadenar una nueva compilación mediante webhooks.



nota

El script de calificación al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se declaran en la especificación del trabajo de laboratorio.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Solucionar problemas de una aplicación mediante la administración del ciclo de vida de una compilación.
- Desencadenar una nueva compilación a través de webhooks.

Andes De Comenzar

Para realizar este ejercicio, necesita acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I y el flujo de imágenes para las aplicaciones Node.js (`nodejs`).
- La aplicación del repositorio Git (`trigger-builds`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab build-app start
```

Requisitos

La aplicación proporcionada se escribe en JavaScript usando el tiempo de ejecución Node.js. Es una aplicación simple basada en el marco (framework) Express. Debe compilar e implementar la aplicación en un clúster OpenShift, según los siguientes requisitos:

- El nombre del proyecto es `youruser-build-app`.
- El nombre de la aplicación es `simple`. Use el script `oc-new-app.sh` del directorio del trabajo de laboratorio para crear e implementar la aplicación. Este script contiene un error intencional que se corrige en un paso posterior. No modifique ni edite el script `oc-new-app.sh`.

- La aplicación implementada se crea a partir del código fuente en el subdirectorio build-app del repositorio Git:

`https://github.com/youruser/D0288-apps.`

- Los módulos NPM requeridos para compilar la aplicación están disponibles en la URL del servidor Nexus:

`http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs.`

Use la variable de entorno `npm_config_registry` para enviar esta información a la imagen del compilador S2I para Node.js.

- Se puede acceder a la aplicación desde la ruta predeterminada:

`simple-youruser-build-app.apps.cluster.domain.example.com.`

Pasos

- Cree el proyecto `youruser-build-app`.
- Ejecute el script `oc-new-app.sh` del directorio `/home/student/D0288/labs/build-app` para crear la aplicación.



Importante

En el script `oc-new-app.sh` existe un error intencional que se corrige en un paso posterior. No modifique ni edite el script `oc-new-app.sh`.

- Verifique si la compilación de la aplicación falla. Establezca el valor correcto para la variable `npm_config_registry` en la configuración de compilación de la aplicación para solucionar el problema.
- Exponga el servicio de aplicación del acceso externo y obtenga la URL de ruta.
- Inicie una nueva compilación y verifique que la aplicación esté lista y en ejecución. Verifique que la aplicación esté accesible con la URL de ruta que obtuvo en el paso anterior.
- Use el webhook genérico para la configuración de compilación para iniciar una nueva compilación de aplicación.
- Califique su trabajo.

Ejecute el siguiente comando en la máquina virtual `workstation` para verificar que se hayan realizado todas las tareas:

```
[student@workstation ~]$ lab build-app grade
```

- Limpie y elimine el proyecto.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-build-app
```

Finalizar

En `workstation`, ejecute el script `lab build-app finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab build-app finish
```

Esto concluye el trabajo de laboratorio.

► Solución

Compilación de aplicaciones

Listado de verificación de rendimiento

En este trabajo de laboratorio, usará OpenShift para administrar las compilaciones de aplicaciones, solucionar problemas de una aplicación y desencadenar una nueva compilación mediante webhooks.



nota

El script de calificación al final de los trabajos de laboratorio del capítulo requiere usar los nombres exactos del proyecto y otros identificadores, como se declaran en la especificación del trabajo de laboratorio.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Solucionar problemas de una aplicación mediante la administración del ciclo de vida de una compilación.
- Desencadenar una nueva compilación a través de webhooks.

Andes De Comenzar

Para realizar este ejercicio, necesita acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I y el flujo de imágenes para las aplicaciones Node.js (nodejs).
- La aplicación del repositorio Git (`trigger-builds`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab build-app start
```

Requisitos

La aplicación proporcionada se escribe en JavaScript usando el tiempo de ejecución Node.js. Es una aplicación simple basada en el marco (framework) Express. Debe compilar e implementar la aplicación en un clúster OpenShift, según los siguientes requisitos:

- El nombre del proyecto es `youruser-build-app`.
- El nombre de la aplicación es `simple`. Use el script `oc-new-app.sh` del directorio del trabajo de laboratorio para crear e implementar la aplicación. Este script contiene un error intencional que se corrige en un paso posterior. No modifique ni edite el script `oc-new-app.sh`.

capítulo 4 | Compilación de aplicaciones

- La aplicación implementada se crea a partir del código fuente en el subdirectorio build-app del repositorio Git:

<https://github.com/youruser/D0288-apps>.

- Los módulos NPM requeridos para compilar la aplicación están disponibles en la URL del servidor Nexus:

[http://\\${RHT_OCP4_NEXUS_SERVER}/repository/nodejs](http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs).

Use la variable de entorno `npm_config_registry` para enviar esta información a la imagen del compilador S2I para Node.js.

- Se puede acceder a la aplicación desde la ruta predeterminada:

`simple-youruser-build-app.apps.cluster.domain.example.com`.

Pasos

1. Cree el proyecto `youruser-build-app`.

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

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

- 1.2. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
```

- 1.3. Cree un nuevo proyecto para alojar la aplicación:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-build-app
```

2. Ejecute el script `oc-new-app.sh` del directorio `/home/student/D0288/labs/build-app` para crear la aplicación.



Importante

En el script `oc-new-app.sh` existe un error intencional que se corrige en un paso posterior. No modifique ni edite el script `oc-new-app.sh`.

- 2.1. Revise el script que crea la aplicación:

```
[student@workstation ~]$ cat ~/D0288/labs/build-app/oc-new-app.sh
...output omitted...
oc new-app --as-deployment-config --name simple --build-env \
  npm_config_registry=http://invalid-server:8081/nexus/content/groups/nodejs \
  https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
  --context-dir build-app
```

- 2.2. Ejecute el script `oc-new-app.sh` para crear la aplicación:

```
[student@workstation ~]$ ~/DO288/labs/build-app/oc-new-app.sh
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
...output omitted...
```

3. Verifique si la compilación de la aplicación falla. Establezca el valor correcto para la variable `npm_config_registry` en la configuración de compilación de la aplicación para solucionar el problema.

- 3.1. Consulte los registros de compilación para identificar el error de compilación. Puede tomar un tiempo para que aparezca un mensaje de error.

```
[student@workstation ~]$ oc logs -f bc/simple
...output omitted...
--> Using 'npm install -s --only=production'
subprocess exited with status 1
subprocess exited with status 1
error: build error: error building at STEP
...output omitted...
```

- 3.2. Una URL no válida del servidor Nexus generó el fallo. Confirme que la URL del servidor Nexus es incorrecta:

```
[student@workstation ~]$ oc set env bc simple --list
# buildconfigs simple
npm_config_registry=http://invalid-server:8081/nexus/content/groups/nodejs
```

- 3.3. Corrija la variable para usar la URL correcta del servidor Nexus:

```
[student@workstation ~]$ oc set env bc simple \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs
buildconfig.build.openshift.io/simple updated
```

Observe que no hay espacio antes ni después del signo igual (=) después de `npm_config_registry`.

- 3.4. Verifique que la variable `npm_config_registry` tenga el valor correcto:

```
[student@workstation ~]$ oc set env bc simple --list
# buildconfigs simple
npm_config_registry=http://nexus-common.../repository/nodejs
```

Observe que no hay espacio antes ni después del signo igual (=) después de `npm_config_registry`. El par key=value completo para la variable de entorno de compilación es demasiado largo para el ancho del papel.

4. Exponga el servicio de aplicación del acceso externo y obtenga la URL de ruta.

- 4.1. Exponga el servicio:

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

4.2. Obtenga la URL de ruta:

```
[student@workstation ~]$ oc get route/simple \
> -o jsonpath='{.spec.host}{"\n"}'
simple-youruser-build-app.apps.cluster.domain.example.com
```

5. Inicie una nueva compilación y verifique que la aplicación esté lista y en ejecución. Verifique que la aplicación esté accesible con la URL de ruta que obtuvo en el paso anterior.

5.1. Inicie una nueva compilación y siga los registros:

```
[student@workstation ~]$ oc start-build simple -F
build.build.openshift.io/simple-2 started
...output omitted...
Push successful
```

5.2. Espere a que la aplicación esté lista y en ejecución:

```
[student@workstation ~]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
simple-1-build  0/1     Error      0          20m
simple-1-cpfqc  1/1     Running    0          4m5s
simple-1-deploy  0/1     Completed   0          4m13s
simple-2-build  0/1     Completed   0          5m41s
```

5.3. Pruebe la aplicación:

```
[student@workstation ~]$ curl \
> simple-${RHT_OCP4_DEV_USER}-build-app.${RHT_OCP4_WILDCARD_DOMAIN}
Simple app for the Building Applications Lab!
```

6. Use el webhook genérico para la configuración de compilación para iniciar una nueva compilación de aplicación.

6.1. Obtenga la URL del webhook genérico que inicia una nueva compilación con el comando `oc describe`.

```
[student@workstation ~]$ oc describe bc simple
Name: simple
...output omitted...
Webhook Generic:
URL: https://apps.cluster.domain.example.com/apis/build.openshift.io/v1/
namespaces/youruser-build-app/buildconfigs/simple/webhooks/<secret>/generic
```

6.2. Obtenga el secreto del webhook ejecutando el comando `oc get bc` y pase la opción `-o json` de volcar los detalles de configuración de compilación en JSON.

```
[student@workstation ~]$ oc get bc simple \
> -o jsonpath=".spec.triggers[*].generic.secret{`\n'}`"
4R8kYYf3014kCSPcECmn
```

Tome nota del secreto para el webhook genérico. Necesitará este secreto en el siguiente paso para desencadenar una nueva compilación.

- 6.3. Inicie una nueva compilación mediante la URL del webhook y el secreto hallados en la salida del paso anterior. Puede ignorar de forma segura el mensaje de error acerca del tipo de contenido no válido en carga útil.

```
[student@workstation ~]$ curl -X POST -k \
> ${RHT_OCP4_MASTER_API}\
> /apis/build.openshift.io/v1/namespaces/${RHT_OCP4_DEV_USER}-build-app\
> /buildconfigs/simple/webhooks/4R8kYYf3014kCSPcECmn/generic
...output omitted...
"status": "Success",
...output omitted...
```

Observe que no haya espacios antes o después de los saltos de línea; la dirección URL completa es demasiado larga para el ancho del papel. Si este comando falla, verifique que RHT_OCP4_MASTER_API no finalice con una barra oblicua.

- 6.4. Enumere todas las compilaciones y verifique que se haya iniciado una nueva compilación:

| [student@workstation ~]\$ oc get builds | | | | | |
|---|--------|-------------|-------------------------|-------------------|--|
| NAME | TYPE | FROM | STATUS | STARTED ... | |
| simple-1 | Source | Git@3e14daf | Failed (AssembleFailed) | About an hour ago | |
| simple-2 | Source | Git@3e14daf | Complete | 20 minutes ago | |
| simple-3 | Source | Git@3e14daf | Complete | 32 seconds ago | |

- 6.5. Espere a que finalice la nueva compilación:

```
[student@workstation ~]$ oc logs -f bc/simple
...output omitted...
Push successful
```

- 6.6. Espere a que la aplicación esté lista y en ejecución:

| [student@workstation ~]\$ oc get pods | | | | |
|---------------------------------------|------------|----------------|----------|-------|
| NAME | READY | STATUS | RESTARTS | AGE |
| simple-1-build | 0/1 | Error | 0 | 29m |
| simple-1-deploy | 0/1 | Completed | 0 | 24m |
| simple-2-build | 0/1 | Completed | 0 | 26m |
| simple-2-c7jvq | 1/1 | Running | 0 | 4m18s |
| simple-2-deploy | 0/1 | Completed | 0 | 4m27s |
| simple-3-build | 0/1 | Completed | 0 | 5m58s |

7. Califique su trabajo.

Ejecute el siguiente comando en la máquina virtual workstation para verificar que se hayan realizado todas las tareas:

```
[student@workstation ~]$ lab build-app grade
```

8. Limpie y elimine el proyecto.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-build-app
```

Finalizar

En workstation, ejecute el script `lab build-app finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab build-app finish
```

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Un recurso `BuildConfig` incluye una estrategia y una o más fuentes de entrada.
- Hay cuatro estrategias de compilación: `Source`, `Pipeline`, `Docker` y `Custom`.
- Estas son las seis fuentes de entrada de compilación, en orden de prioridad: `Dockerfile`, `Git`, `image`, `binary`, `input secrets` y `external artifacts`.
- Gestione el ciclo de vida de una compilación con los comandos CLI `oc`, como `oc start-build`, `oc cancel-build`, `oc delete`, `oc describe` y `oc logs`.
- Las compilaciones pueden iniciarse automáticamente a través de desencadenadores de compilación, como un desencadenador de cambio de imagen y `webhook`.
- Puede realizar tareas de validación durante las compilaciones mediante un hook de compilación `post-commit` (posterior a la confirmación).

capítulo 5

Personalización de compilaciones de fuente a imagen

Meta

Personalizar una imagen de compilador S2I existente y crear una nueva.

Objetivos

- Describir los pasos requeridos y opcionales en el proceso de compilación de fuente a imagen.
- Personalizar una imagen de compilador S2I existente con scripts.
- Crear una nueva imagen de compilador S2I con herramientas de S2I.

Secciones

- Descripción de la arquitectura de fuente a imagen (y cuestionario)
- Personalización de una imagen de compilador S2I existente (y ejercicio guiado)
- Creación de una imagen de compilador S2I (y ejercicio guiado)

Trabajo de laboratorio

Personalización de compilaciones de fuente a imagen

Descripción de la arquitectura de fuente a imagen

Objetivos

Después de completar esta sección, deberá ser capaz de describir los pasos requeridos y opcionales en una compilación de fuente a imagen.

Detección de lenguaje de fuente a imagen (S2I)

OpenShift puede crear aplicaciones directamente del código fuente almacenado en un repositorio Git. La sintaxis más simple para el comando `oc new-app` toma solo la URL del repositorio Git como argumento y, luego, OpenShift intenta detectar automáticamente el lenguaje de programación usado por la aplicación y seleccionar una imagen de compilador compatible.

La lógica de la detección automática no es perfecta. El comando `oc new-app` puede usar una gama de opciones de línea de comandos para forzar una opción particular. Estas opciones de la línea de comandos se presentaron anteriormente en este curso.

La característica de detección del lenguaje de programación se basa en buscar nombres de archivos específicos en la raíz del repositorio Git. En la tabla siguiente se muestran algunas de las opciones más comunes, pero no se trata de una extensa lista de todos los idiomas compatibles con el origen a la imagen. Consulte los documentos del producto para ver todas las reglas para cada versión de Red Hat OpenShift Container Platform:

| Archivos | Compilador de lenguaje | Lenguaje de programación |
|--------------------------|------------------------|------------------------------------|
| Dockerfile | N/D | Compilación de Dockerfile (no S2I) |
| pom.xml | jee | Java (con JBoss EAP) |
| app.json, package.json | nodejs | Node.js (JavaScript) |
| composer.json, index.php | php | PHP |

OpenShift sigue un algoritmo de varios pasos para determinar si la URL apunta a un repositorio de código fuente y, si ese es el caso, cuál es la imagen de compilador que debe realizar la compilación. A continuación se muestra una descripción simplificada del algoritmo:

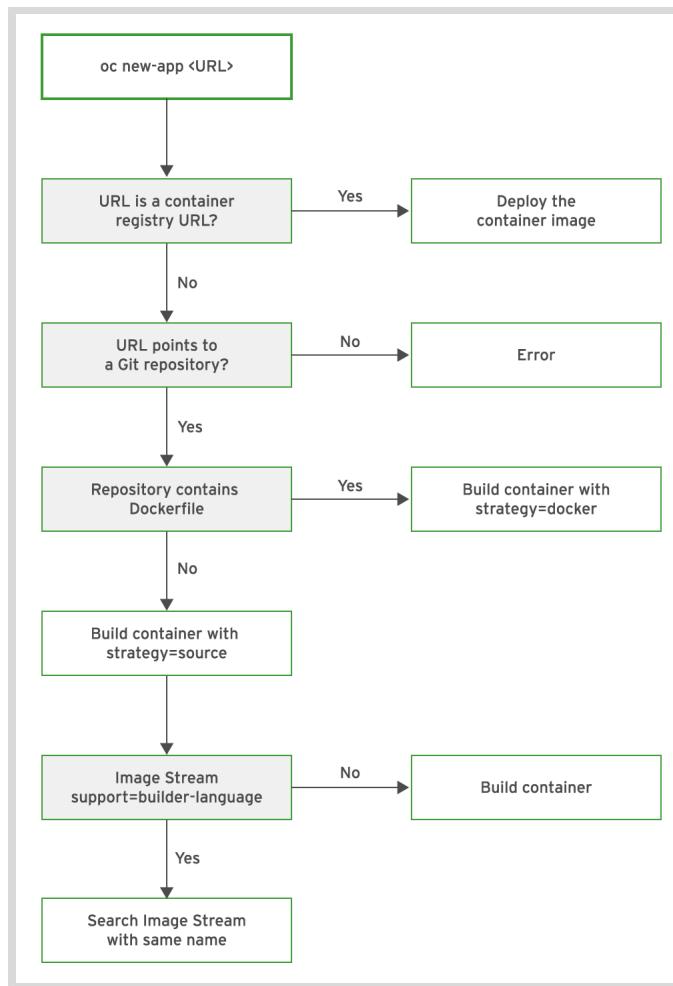


Figura 5.1: Selección de procesos de compilación S2I

1. OpenShift accede a la URL como una URL de registro de contenedor. Si esto se realiza correctamente, no necesita crear una configuración de compilación. OpenShift crea la configuración de implementación y otros recursos que se requieren para implementar una imagen de contenedor.
2. Si la URL apunta a un repositorio Git, OpenShift recupera una lista de archivos del repositorio y busca un archivo Dockerfile. Si lo encuentra, la configuración de compilación usa una estrategia Docker. De lo contrario, la configuración de compilación usa una estrategia Source, que necesita una imagen de compilador S2I.
3. OpenShift busca flujos de imágenes que coincidan con el nombre del compilador como el valor de la anotación `support`s. La primera coincidencia se convierte en la imagen de compilador S2I.
4. Si ninguna anotación coincide, OpenShift busca un flujo de imágenes cuyo nombre coincida con el nombre del compilador de lenguaje. La primera coincidencia se convierte en la imagen de compilador S2I.

Los pasos 3 y 4 facilitan la adición de nuevas imágenes de compilador a un clúster OpenShift. También significa que varias imágenes de compilador pueden ser posibles coincidencias. Anteriormente en este capítulo, hablamos sobre las opciones de la línea de comandos `oc new-app` para evitar dicha ambigüedad y para garantizar que OpenShift seleccione el flujo de imágenes correcto como la imagen de compilador S2I.

capítulo 5 | Personalización de compilaciones de fuente a imagen

Por ejemplo, al ejecutar el siguiente comando, el comando oc identifica que hace referencia a una URL de registro e inicia la implementación del contenedor:

```
[user@host ~]$ oc new-app --as-deployment-config  
registry.access.redhat.com/ubi8/ubi:8.0
```

Como alternativa, al ejecutar el siguiente comando, el comando oc identifica que está usando un repositorio Git y va a clonar el repositorio para buscar un archivo Dockerfile. Si el clúster OpenShift encuentra un archivo Dockerfile en el directorio raíz del repositorio, desencadena un nuevo proceso de compilación de contenedor.

```
[user@host ~]$ oc new-app --as-deployment-config \  
> https://github.com/RedHatTraining/D0288-apps/ubi-echo
```

Si el clúster OpenShift encuentra uno de los archivos mencionados en la tabla anterior en el directorio raíz del repositorio, el clúster OpenShift inicia un proceso S2I usando su respectivo compilador de imágenes.

Para forzar el uso de un determinado flujo de imágenes, puede usar la opción -i para una aplicación PHP 7.3:

```
[user@host ~]$ oc new-app --as-deployment-config -i php:7.3 \  
> https://github.com/RedHatTraining/D0288-apps/php-helloworld
```

El clúster OpenShift busca un flujo de imágenes denominado php y busca la versión 7.3 para invocar el compilador.

El proceso de compilación de Fuente a imagen (Source-to-Image, S2I)

El proceso de compilación S2I involucra tres componentes fundamentales, que se combinan para crear una imagen de contenedor final:

Código fuente de la aplicación

Este es el código fuente para la aplicación.

Scripts S2I

Los scripts S2I son un conjunto de scripts que ejecuta el proceso de compilación de OpenShift para personalizar la imagen de compilador S2I. Los scripts S2I se pueden escribir en cualquier lenguaje de programación, siempre y cuando se puedan ejecutar dentro de la imagen de compilador S2I.

Imagen de compilador S2I

Esta es una imagen de un contenedor que incluye el entorno de tiempo de ejecución requerido para la aplicación. Por lo general, contiene compiladores, intérpretes, scripts y otras dependencias que son necesarias para ejecutar la aplicación.

El proceso de compilación S2I depende de algunos scripts S2I, que ejecuta en diversas etapas del flujo de trabajo de compilación. En la siguiente tabla, se muestran los scripts y una breve descripción de lo que hacen:

| Script | ¿Obligatorio? | Descripción |
|----------------|---------------|---|
| assemble | Sí | El script <code>assemble</code> compila la aplicación de la fuente y la coloca en los directorios adecuados dentro de la imagen. |
| run | Sí | El script <code>run</code> ejecuta su aplicación. Se recomienda usar el comando <code>exec</code> al ejecutar cualquier proceso de contenedor en el script <code>run</code> . Esto garantiza la propagación de la señal y el apagado correcto de cualquier proceso iniciado por el script <code>run</code> . |
| save-artifacts | No | El script <code>save-artifacts</code> recopila todas las dependencias requeridas para la aplicación y las guarda para acelerar las compilaciones posteriores. Por ejemplo, para Ruby, gems instalados por Bundler, o para Java contenido .m2. Esto significa que la compilación no tiene que volver a descargar este contenido, lo que mejora el tiempo de compilación. Estas dependencias se recopilan en un archivo tar y se transmiten a la salida estándar. |
| usage | No | El script <code>usage</code> proporciona una descripción de cómo usar adecuadamente su imagen. |
| test/run | No | El script <code>test/run</code> le permite crear un proceso simple para verificar si la imagen está funcionando correctamente. |

Flujo de trabajo de compilación S2I

El proceso de compilación S2I es de la siguiente manera:

1. OpenShift instancia un contenedor basado en la imagen de compilador S2I y, luego, crea un archivo tar que contiene el código fuente y los scripts S2I. Luego, OpenShift envía el archivo tar al contenedor.
2. Antes de ejecutar el script `assemble`, OpenShift extrae el archivo tar del paso anterior y guarda el contenido en la ubicación especificada por la opción `--destination` o por la etiqueta `io.openshift.s2i.destination` de la imagen de compilador. La ubicación predeterminada es el directorio `/tmp`.
3. Si esta es una compilación incremental, el script `assemble` restaura los artefactos de compilación que el script `save-artifacts` guardó anteriormente en un archivo tar.
4. El script `assemble` compila la aplicación de la fuente y coloca los binarios en los directorios adecuados.
5. Si esta es una compilación incremental, se ejecuta el script `save-artifacts` y este guarda todos los artefactos de compilación de las dependencias en un archivo tar.
6. Después de que finaliza el script `assemble`, el contenedor debe crear la imagen final, y OpenShift define el script `run` como la instrucción CMD para la imagen final.

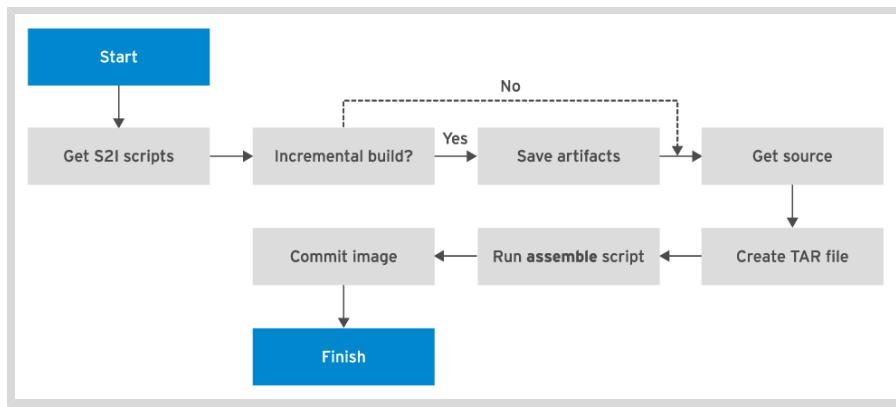


Figura 5.2: Flujo de trabajo de compilación S2I

Para crear una imagen de compilador, declare un archivo `Dockerfile` con las herramientas necesarias para crear la aplicación, como compiladores, herramientas de compilación y todos los archivos de script mencionados anteriormente. El siguiente archivo de compilador `Dockerfile` define un compilador del servidor NGINX:

```

FROM registry.access.redhat.com/ubi8/ubi:8.0

LABEL io.k8s.description="My custom Builder" \①
      io.k8s.display-name="Nginx 1.6.3" \
      io.openshift.expose-services="8080:http" \
      io.openshift.tags="builder,webserver,html,nginx" \
      io.openshift.s2i.scripts-url="image:///usr/libexec/s2i" ②

RUN yum install -y epel-release && \③
    yum install -y --nодocs nginx && \
    yum clean all

EXPOSE 8080④

COPY ./s2i/bin/ /usr/libexec/s2i⑤
  
```

- ① Establezca las etiquetas que se usan para OpenShift para describir la imagen de compilador.
- ② Indique a S2I dónde encontrar sus scripts obligatorios (`run`, `assemble`).
- ③ Instale el paquete del servidor web NGINX y limpie la memoria caché de Yum.
- ④ Establezca el puerto predeterminado para las aplicaciones creadas con esta imagen.
- ⑤ Copie los scripts S2I en el directorio `/usr/libexec/s2i`.

El script de ensamblaje se puede definir de la siguiente manera:

```

#!/bin/bash -e

if [ "$(ls -A /tmp/src)" ]; then
    mv /tmp/src/* /usr/share/nginx/html/①
fi
  
```

- ① Anule el archivo `index.html` de NGINX predeterminado.

```
#!/bin/bash -e
/usr/sbin/nginx -g "daemon off;"①
```

- ① Evite que el proceso NGINX se ejecute como un daemon para que el contenedor no se cierre después de que el proceso ejecute el script exec.

Anulación de los scripts de compilador S2I

Las imágenes de compilador S2I proporcionan sus propios scripts S2I predeterminados. Puede sobrescribir los scripts S2I predeterminados para cambiar la manera en que se compila y se ejecuta su aplicación. Puede sobrescribir el comportamiento de compilación predeterminado sin la necesidad de crear una nueva imagen de compilador S2I mediante la bifurcación del código fuente para el compilador S2I original.

La manera más simple de sobrescribir los scripts S2I predeterminados para una aplicación es incluir sus scripts S2I en el repositorio del código fuente para su aplicación. Puede proporcionar scripts S2I en la carpeta `.s2i/bin` del repositorio del código fuente de la aplicación.

Cuando OpenShift inicia el proceso S2I, inspecciona la carpeta de código fuente, los scripts S2I personalizados y el código fuente de la aplicación. OpenShift incluye todos estos archivos en el archivo tar injectado en la imagen de compilador S2I. Luego, OpenShift ejecuta el script personalizado `assemble` en lugar del script predeterminado `assemble` que está incluido en el compilador S2I, seguido por los otros scripts personalizados anulados (si corresponde).



Referencias

Cómo sobrescribir scripts de compilador S2I

<https://blog.openshift.com/override-s2i-builder-scripts/>

Cómo crear una imagen de compilador S2I

<https://blog.openshift.com/create-s2i-builder-image/>

Herramienta de Fuente a imagen (Source-to-Image, S2I)

<https://github.com/openshift/source-to-image>

Interfaz de línea de comandos S2I

<https://github.com/openshift/source-to-image/blob/master/docs/cli.md>

Para obtener más información acerca de las variables del entorno de compilación de las imágenes de compilador S2I de OpenShift estándares, consulte la guía *Imágenes*, en la documentación para Red Hat OpenShift Container Platform 4.5 en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/images/

► Cuestionario

Descripción de la arquitectura de fuente a imagen

Elija las respuestas correctas para las siguientes preguntas:

- ▶ 1. ¿Cuál de los siguientes scripts S2I es responsable de compilar los binarios de la aplicación en una compilación S2I?
 - a. run
 - b. assemble
 - c. save-artifacts
 - d. test/run

- ▶ 2. ¿Cuáles dos de los siguientes enunciados acerca del proceso de compilación S2I son verdaderos? (Elija dos opciones).
 - a. El script assemble se ejecuta antes de que se extraiga el archivo tar que contiene el código fuente de la aplicación y los scripts S2I.
 - b. El script assemble se ejecuta después de que se extrae el archivo tar que contiene el código fuente de la aplicación y los scripts S2I.
 - c. El script run se ejecuta después de que se extrae el archivo tar que contiene el código fuente de la aplicación y los scripts S2I.
 - d. El script run está definido como la instrucción CMD para la imagen de contenedor final.
 - e. El script assemble está definido como la instrucción CMD para la imagen de contenedor final.

- ▶ 3. ¿En cuál de los siguientes directorios (en relación con la raíz del código fuente) proporcionaría sus propios scripts S2I personalizados?
 - a. .scripts/bin
 - b. .s2i/bin
 - c. etc/bin
 - d. usr/local/bin

- ▶ 4. ¿Cuáles dos de los siguientes scripts son obligatorios en una compilación de S2I? (Elija dos opciones).
 - a. usage
 - b. test/run
 - c. assemble
 - d. run
 - e. save-artifacts

► **5. ¿Cuáles dos de los siguientes escenarios son buenos candidatos para compilaciones S2I incrementales? (Elija dos opciones).**

- a. Una aplicación Java EE con una gran cantidad de dependencias JAR administradas con Apache Maven.
- b. Una aplicación que depende de un archivo de volcado de base de datos SQL que se invoca en el inicio de la aplicación.
- c. Una aplicación web Ruby que tiene una gran cantidad de activos estáticos, como imágenes, CSS y archivos HTML.
- d. Una aplicación Node.js con dependencias administradas con npm.

► **6. ¿Cuál de las siguientes etiquetas indica a la imagen de compilador S2I el directorio donde se almacenan los scripts?**

- a. io.openshift.s2i.scripts-url
- b. io.openshift.s2i.scripts-dir
- c. io.openshift.s2i.scripts-directory
- d. io.openshift.s2i.scripts-URL

► Solución

Descripción de la arquitectura de fuente a imagen

Elija las respuestas correctas para las siguientes preguntas:

► 1. **¿Cuál de los siguientes scripts S2I es responsable de compilar los binarios de la aplicación en una compilación S2I?**

- a. run
- b. assemble
- c. save-artifacts
- d. test/run

► 2. **¿Cuáles dos de los siguientes enunciados acerca del proceso de compilación S2I son verdaderos? (Elija dos opciones).**

- a. El script assemble se ejecuta antes de que se extraiga el archivo tar que contiene el código fuente de la aplicación y los scripts S2I.
- b. El script assemble se ejecuta después de que se extrae el archivo tar que contiene el código fuente de la aplicación y los scripts S2I.
- c. El script run se ejecuta después de que se extrae el archivo tar que contiene el código fuente de la aplicación y los scripts S2I.
- d. El script run está definido como la instrucción CMD para la imagen de contenedor final.
- e. El script assemble está definido como la instrucción CMD para la imagen de contenedor final.

► 3. **¿En cuál de los siguientes directorios (en relación con la raíz del código fuente) proporcionaría sus propios scripts S2I personalizados?**

- a. .scripts/bin
- b. .s2i/bin
- c. etc/bin
- d. usr/local/bin

► 4. **¿Cuáles dos de los siguientes scripts son obligatorios en una compilación de S2I? (Elija dos opciones).**

- a. usage
- b. test/run
- c. assemble
- d. run
- e. save-artifacts

► **5. ¿Cuáles dos de los siguientes escenarios son buenos candidatos para compilaciones S2I incrementales? (Elija dos opciones).**

- a. Una aplicación Java EE con una gran cantidad de dependencias JAR administradas con Apache Maven.
- b. Una aplicación que depende de un archivo de volcado de base de datos SQL que se invoca en el inicio de la aplicación.
- c. Una aplicación web Ruby que tiene una gran cantidad de activos estáticos, como imágenes, CSS y archivos HTML.
- d. Una aplicación Node.js con dependencias administradas con npm.

► **6. ¿Cuál de las siguientes etiquetas indica a la imagen de compilador S2I el directorio donde se almacenan los scripts?**

- a. io.openshift.s2i.scripts-url
- b. io.openshift.s2i.scripts-dir
- c. io.openshift.s2i.scripts-directory
- d. io.openshift.s2i.scripts-URL

Personalización de una imagen base S2I existente

Objetivos

Tras finalizar esta sección, deberá ser capaz de personalizar los scripts S2I de una imagen de compilador S2I existente.

Personalización de scripts de una imagen de compilador S2I

Los scripts S2I están empaquetados dentro de las imágenes del compilador S2I de manera predeterminada. En ciertos escenarios, es posible que desee personalizar estos scripts para cambiar la manera en que se compila y se ejecuta su aplicación, sin volver a compilar la imagen.

El proceso de compilación de S2I proporciona un método para sobrescribir los scripts S2I predeterminados. Puede proporcionar sus propios scripts S2I en la carpeta `.s2i/bin` del código fuente de la aplicación. Durante el proceso de compilación, se detectan y se ejecutan automáticamente los scripts S2I personalizados en lugar de los scripts S2I predeterminados que están incluidos en la imagen de compilador.

Según la cantidad de personalización que necesite aplicarse en los scripts anulados, puede elegir reemplazar por completo los scripts S2I predeterminados con su propia versión. Como alternativa, puede crear un script *wrapper* que invoque los scripts predeterminados y, a continuación, agregar la personalización necesaria antes o después de la invocación.

Por ejemplo, supongamos que desea personalizar los scripts S2I para la imagen de compilador S2I `rhsc1/php-73-rhel7` y cambiar la manera en que se compila y se ejecuta la aplicación. Puede usar el siguiente procedimiento para personalizar los scripts S2I proporcionados en esta imagen de compilador:

- Use el comando `podman pull` para extraer la imagen de contenedor desde el registro de contenedor al sistema local. Use el comando `sudo podman inspect` para obtener el valor de la etiqueta `io.openshift.s2i.scripts-url`, para determinar la ubicación predeterminada de los scripts S2I en la imagen.

```
[user@host ~]$ sudo podman pull \
> myregistry.example.com/rhsc1/php-73-rhel7
...
Digest: sha256:...
[user@host ~]$ sudo podman inspect --format='{{ index .Config.Labels
"io.openshift.s2i.scripts-url"}}' rhsc1/php-73-rhel7
image:///usr/libexec/s2i
```

- También puede usar el comando `skopeo inspect` para recuperar la misma información directamente de un registro remoto:

```
[user@host ~]$ skopeo inspect \
> docker://myregistry.example.com/rhscl/php-73-rhel7 \
> | grep io.openshift.s2i.scripts-url
"io.openshift.s2i.scripts-url": "image:///usr/libexec/s2i",
```

- Cree un contenedor (wrapper) para el script `assemble` en la carpeta `.s2i/bin`:

```
#!/bin/bash
echo "Making pre-invocation changes..."

/usr/libexec/s2i/assemble
rc=$?

if [ $rc -eq 0 ]; then
    echo "Making post-invocation changes..."
else
    echo "Error: assemble failed!"
fi

exit $rc
```

- De manera similar, cree un contenedor (wrapper) para el script `run` en la carpeta `.s2i/bin`:

```
#!/bin/bash
echo "Before calling run..."
exec /usr/libexec/s2i/run
```



nota

Cuando ajusta el script `run`, debe usar `exec` para invocarlo. Esto garantiza que el script `run` predeterminado aún se ejecute con un ID de proceso de 1. Si no hacerlo genera errores de propagación de señal durante el apagado, y es posible que su aplicación no funcione correctamente. Esto también implica que no pueda ejecutar comandos adicionales en el script del contenedor (wrapper) después de invocar el script `run` predeterminado.

Compilaciones incrementales en S2I

Cuando compila aplicaciones en un clúster OpenShift con S2I, es muy común compilar, implementar y probar pequeños cambios incrementales en sus aplicaciones. Ciertas organizaciones han adoptado técnicas de *integración continua* (continuous integration, CI) y *entrega continua* (continuous delivery, CD), donde la aplicación se compila e implementa varias veces en un ciclo iterativo rápido que a menudo no requiere intervención manual.

Cuando su aplicación se compila de manera modular con varios componentes y bibliotecas dependientes, las compilaciones de S2I requieren mucho tiempo debido a la naturaleza inmutable de los contenedores. El proceso de compilación debe buscar las dependencias y, luego, compilar e implementar la aplicación cada vez que haya un cambio en el código fuente.

El proceso de compilación de S2I proporciona un mecanismo para reducir el tiempo de compilación mediante la invocación del script `save-artifacts` después de invocar el script `assemble`, como parte del ciclo de vida S2I. El script `save-artifacts` garantiza que los

artefactos dependientes (librerías y componentes requeridos para la aplicación) se guarden para futuras compilaciones.

Durante la siguiente compilación, el script `assemble` restaura los artefactos almacenados en caché antes de compilar la aplicación a partir del código fuente. Tenga en cuenta que el script `save-artifacts` es responsable de transmitir dependencias a `stdout` en un archivo tar.



Importante

La salida del script `save-artifacts` solo debe incluir la salida del flujo de tar (y nada más). Debe redireccionar la salida de otros comandos en el script a `/dev/null`.

Por ejemplo, supongamos que está desarrollando una aplicación basada en Java EE con muchas dependencias administradas con Apache Maven. Suponiendo que ha compilado una imagen de compilador S2I, donde el script `assemble` compila y empaqueta el archivo JAR de la aplicación, las compilaciones incrementales que pueden reutilizar archivos JAR previamente descargados reducen el tiempo de compilación por un amplio margen. Apache Maven almacena las dependencias JAR en la carpeta `$HOME/.m2`.

El script `save-artifacts` que almacena en caché los archivos JAR de Maven se puede describir de la siguiente manera:

```
#!/bin/sh -e

# Stream the .m2 folder tar archive to stdout
if [ -d ${HOME}/.m2 ]; then
    pushd ${HOME} > /dev/null
    tar cf - .m2
    popd > /dev/null
fi
```

El código correspondiente para restaurar los artefactos antes de la compilación está en el script `assemble`:

```
# Restore the .m2 folder
...
if [ -d /tmp/artifacts/.m2 ]; then
    echo "---> Restoring maven artifacts..."
    mv /tmp/artifacts/.m2 ${HOME}/
fi
...
```



Referencias

Puede obtener más información en el capítulo *Creación de imágenes* de la guía *Imágenes de Red Hat OpenShift Container Platform 4.5*; en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/images/index#creating-images

Cómo sobrescribir scripts de compilador S2I

<https://blog.openshift.com/override-s2i-builder-scripts>

► Ejercicio Guiado

Personalización de compilaciones S2I

En este ejercicio, personalizará los scripts de S2I de una imagen de compilador S2I para agregar una página de información a la aplicación.

Resultados

Deberá ser capaz de personalizar los scripts `assemble` y `run` de una imagen de compilador del servidor HTTP de Apache. Sobrescribirá los scripts incorporados predeterminados con sus propias versiones personalizadas.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen de compilador S2I del servidor HTTP de Apache (`rhscl/httpd-24-rhel7`).
- Una bifurcación del repositorio Git que contiene el código fuente de la aplicación (`s2i-scripts`).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos del ejercicio y descargar los archivos de trabajos de laboratorio y soluciones:

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

► 1. Explore los scripts S2I incluidos en la imagen del compilador `rhscl/httpd-24-rhel7`.

- 1.1. En la máquina virtual `workstation`, ejecute la imagen `rhscl/httpd-24-rhel7` desde una ventana de terminal, y sobrescriba el punto de entrada del contenedor para ejecutar una shell:

```
[student@workstation ~]$ sudo podman run --name test -it rhscl/httpd-24-rhel7 bash
Trying to pull registry.access.redhat.com/rhscl/httpd-24-rhel7:latest...Getting
image source signatures
...output omitted...
bash-4.2$
```

- 1.2. Inspeccione los scripts S2I incluidos en la imagen del compilador. Los scripts S2I están ubicados en la carpeta `/usr/libexec/s2i`:

```
bash-4.2$ cat /usr/libexec/s2i/assemble
...output omitted...
bash-4.2$ cat /usr/libexec/s2i/run
...output omitted...
bash-4.2$ cat /usr/libexec/s2i/usage
...output omitted...
```

Salga del contenedor:

```
bash-4.2$ exit
```

► 2. Revise el código fuente de la aplicación con los scripts S2I personalizados.

- 2.1. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación master del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 2.2. Inspeccione el archivo /home/student/D0288-apps/s2i-scripts/index.html.

El archivo HTML contiene un mensaje simple:

```
Hello Class! D0288 rocks!!!
```

- 2.3. Los scripts S2I personalizados están ubicados en la carpeta /home/student/D0288-apps/s2i-scripts/.s2i/bin. El script .s2i/bin/assemble copia el archivo index.html de la fuente de la aplicación a la raíz del documento del servidor web en /opt/app-root/src. También crea un archivo info.html que contiene información del entorno y del tiempo de compilación de la página:

```
...output omitted...
#####
# CUSTOMIZATION STARTS HERE #####
echo "----> Installing application source"
cp -Rf /tmp/src/*.html .

DATE=`date "+%b %d, %Y @ %H:%M %p"`

echo "----> Creating info page"
echo "Page built on $DATE" >> ./info.html
echo "Proudly served by Apache HTTP Server version $HTTPD_VERSION" >> ./info.html

#####
# CUSTOMIZATION ENDS HERE #####
...output omitted...
```

- 2.4. El script .s2i/bin/run cambia el nivel del registro predeterminado de los mensajes de inicio en el servidor web por debug:

```
# Make Apache show 'debug' level logs during startup
exec run-httpd -e debug $@
```

► 3. Implemente la aplicación en un clúster OpenShift. Verifique que se ejecuten los scripts S2I personalizados.

- 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 D0288-apps]$ source /usr/local/etc/ocp4.config
```

- 3.2. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation D0288-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 para la aplicación. Coloque como prefijo del nombre del proyecto su nombre de usuario de desarrollador.

```
[student@workstation D0288-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-s2i-scripts
Now using project "youruser-s2i-scripts" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 3.4. Cree una nueva aplicación denominada `bonjour` a partir de las fuentes en Git. Necesita anteponer la URL de Git con el flujo de imágenes `httpd:2.4`, con una tilde (~), para asegurarse de que la aplicación use la imagen de compilador `rhscl/httpd24-rhel7`.

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config \
> --name bonjour \
> httpd:2.4-https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
> --context-dir s2i-scripts
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "bonjour" created
buildconfig.build.openshift.io "bonjour" created
deploymentconfig.apps.openshift.io "bonjour" created
service "bonjour" created
--> Success
...output omitted...
```

- 3.5. Vea los registros de compilación. Espere hasta que termine la compilación y la imagen del contenedor de aplicaciones se envíe al registro OpenShift:

```
[student@workstation D0288-apps]$ oc logs -f bc/bonjour
Cloning "https://github.com/youruser/D0288-apps" ...
...output omitted...
---> Enabling s2i support in httpd24 image
AllowOverride All
---> Installing application source
---> Creating info page
```

```
Pushing image image-registry.openshift-image-registry.svc:5000/youruser-s2i-
scripts/bonjour:latest ... ...
...output omitted...
Push successful
```

Observe que se ejecutan los scripts S2I personalizados proporcionados por la aplicación en lugar de los scripts S2I incorporados de la imagen del compilador.

► 4. Pruebe la aplicación.

- 4.1. Espere a que se implemente la aplicación y, luego, consulte el estado del pod de la aplicación. El pod de la aplicación debe estar en el estado **Running** (En ejecución):

```
[student@workstation D0288-apps]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
bonjour-1-build  0/1     Completed   0          2m3s
bonjour-1-deploy 0/1     Completed   0          66s
bonjour-1-km4bq  1/1     Running    0          58s
```

- 4.2. Exponga la aplicación a un acceso externo usando una ruta.

```
[student@workstation D0288-apps]$ oc expose svc bonjour
route.route.openshift.io/bonjour exposed
```

- 4.3. Obtenga la URL de enrutamiento con el comando `oc get route`:

```
[student@workstation D0288-apps]$ oc get route
NAME      HOST/PORT
bonjour   bonjour-youruser-s2i-scripts.apps.cluster.domain.example.com ...
```

- 4.4. Invoque la página de índice de la aplicación con el comando `curl` y la URL de la ruta del comando anterior:

```
[student@workstation D0288-apps]$ curl \
> http://bonjour-${RHT_OCP4_DEV_USER}-s2i-scripts.${RHT_OCP4_WILDCARD_DOMAIN}
Hello Class! D0288 rocks!!!
```

Debe ver el contenido del archivo `index.html` en la fuente de la aplicación.

- 4.5. Invoque la página de información de la aplicación con el comando `curl`:

```
[student@workstation D0288-apps]$ curl \
> http://bonjour-${RHT_OCP4_DEV_USER}-s2i-scripts.${RHT_OCP4_WILDCARD_DOMAIN}\
> /info.html
Page built on Jun 11, 2019 @ 16:12 PM
Proudly served by Apache HTTP Server version 2.4
```

Debe ver el contenido del archivo `info.html` con detalles acerca de cuándo se compiló la página y la versión del servidor HTTP de Apache.

- 4.6. Inspeccione los registros para el pod de la aplicación. Recuerde que el nivel del registro de inicio se cambió a `debug` en el script `run`. Debe ver los mensajes del registro de nivel `debug` en el inicio:

```
[student@workstation DO288-apps]$ oc logs bonjour-1-km4bq
...output omitted...
[Fri Nov 03 16:12:21.690941 2017] [so:debug] [pid 9] mod_so.c(266): AH01575:
loaded module systemd_module from /opt/rh/httpd24/root/etc/httpd/modules/
mod_systemd.so
[Fri Nov 03 16:12:21.691050 2017] [so:debug] [pid 9] mod_so.c(266): AH01575:
loaded module cgi_module from /opt/rh/httpd24/root/etc/httpd/modules/mod_cgi.so
...output omitted...
[Fri Nov 03 16:12:21.742471 2017] [ssl:debug] [pid 9] ssl_engine_init.c(270):
AH01886: SSL FIPS mode disabled
...output omitted...
[Fri Nov 03 16:12:21.745520 2017] [mpm_prefork:notice] [pid 9] AH00163:
Apache/2.4.25 (Red Hat) OpenSSL/1.0.1e-fips configured -- resuming normal
operations
[Fri Nov 03 16:12:21.745530 2017] [core:notice] [pid 9] AH00094: Command line:
'httpd -D FOREGROUND -e debug'
...output omitted...
10.131.0.16 - - [03/Nov/2019:16:28:53 +0000] "GET / HTTP/1.1" 200 28 "-"
"curl/7.29.0"
10.128.2.216 - - [03/Nov/2019:16:29:03 +0000] "GET /info.html HTTP/1.1" 200 87 "-"
"curl/7.29.0"
```

- 5. Realice una limpieza. Elimine el proyecto:

```
[student@workstation DO288-apps]$ cd ~
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-s2i-scripts
```

- 6. Elimine el contenedor test que creó anteriormente para ver los scripts S2I predeterminados.

```
[student@workstation ~]$ sudo podman rm test
```

Finalizar

En la máquina virtual **workstation**, ejecute el script **lab s2i-scripts finish** para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab s2i-scripts finish
```

Esto concluye el ejercicio guiado.

Creación de una imagen de compilador S2I

Objetivos

Tras finalizar esta sección, deberá ser capaz de crear una imagen de compilador S2I con la herramienta de línea de comandos s2i.

Compilación y publicación de una imagen de compilador S2I personalizada

El proceso de compilación S2I combina el código fuente de la aplicación con una *imagen de compilador S2I* adecuada para producir la imagen de contenedor de aplicaciones final que se implementa en un clúster OpenShift.

Antes de implementar la imagen de compilador S2I en un clúster OpenShift, donde pueden usarla otros desarrolladores para compilar aplicaciones, es importante compilar y probar la imagen con la herramienta de la línea de comandos s2i. Instale la herramienta s2i en el equipo local para crear y probar las imágenes de compilador S2I fuera de un clúster OpenShift.

Instalación de la herramienta S2I

En sistemas RHEL 7, la herramienta s2i está disponible en el paquete *source-to-image* y se puede instalar con Yum. Asegúrese de que su sistema se haya suscrito al repositorio de Yum `rhel-server-rhscl-7-rpms` y lo haya habilitado.

Para otros sistemas operativos, la herramienta s2i puede descargarse de la página del proyecto *source-to-image upstream* en:

<https://github.com/openshift/source-to-image/releases>

Uso de la herramienta S2I

Use el comando `s2i create` para crear los archivos de plantilla requeridos para crear una nueva imagen de compilador S2I:

```
[user@host ~]$ s2i create image_name directory
```

El comando anterior crea una carpeta denominada `directory` y la completa con los siguientes archivos de plantilla que puede actualizar según sea necesario:

```
directory
├── Dockerfile ①
├── Makefile
├── README.md
└── s2i ②
    └── bin
        ├── assemble
        ├── run
        ├── save-artifacts
        └── usage
```

```

└── test
    ├── run
    └── test-app ③
        └── index.html

```

- ① El Dockerfile para la imagen de compilador S2I
- ② El directorio de scripts S2I
- ③ Carpeta para copiar el código fuente de su aplicación para pruebas locales

El comando `s2i create` crea un Dockerfile plantilla con comentarios, personalizado para ejecutarlo en un clúster OpenShift con un ID de usuario aleatorio y etiquetas específicas de OpenShift. También crea series para los scripts S2I que se pueden personalizar para satisfacer las necesidades de su aplicación. Después de que actualice el Dockerfile y los scripts S2I según sea necesario, puede compilar la imagen del compilador con el comando `podman build`.

```
[user@host ~]$ sudo podman build -t builder_image .
```

Cuando la imagen del compilador esté lista, puede compilar una imagen de contenedor de aplicaciones con el comando `s2i build`. Esto le permite probar la imagen del compilador S2I localmente, sin la necesidad de enviarla a un servidor de registro e implementar una aplicación con la imagen de compilador en un clúster OpenShift:

```
[user@host ~]$ s2i build src builder_image tag_name
```



nota

Si incluye instrucciones que no forman parte de la especificación OCI, como la instrucción `ONBUILD` en el Dockerfile de la imagen de compilador, asegúrese de usar la opción `--format docker` con el comando `podman build` al compilar la imagen de compilador S2I. Esta opción reemplaza el formato `oci` predeterminado usado por Podman.

El comando `s2i build` combina el código fuente de la aplicación definido en la opción `src` con la imagen de contenedor `builder_image` para producir la imagen de contenedor de aplicaciones etiquetada con una etiqueta de `tag_name`. Este comando emula el proceso S2I que usa el clúster OpenShift, mediante la inyección automática del código fuente proporcionado en la imagen del compilador, pero usa el servicio Docker local para compilar una imagen de contenedor de prueba, en lugar de implementar la imagen del clúster OpenShift.

Ejecute la imagen con el comando `podman run` para probar la imagen de contenedor de aplicaciones producida por el comando `s2i build`. Tenga en cuenta que si la imagen de contenedor de aplicaciones se implementará en OpenShift, debe simular la ejecución del contenedor con un ID de usuario aleatorio mediante el uso del indicador `-u` para el comando `podman run`.

**Importante**

El comando `s2i build` requiere el uso de un servicio Docker local ya que usa la API de Docker directamente a través del socket para compilar la imagen de contenedor S2I. En entornos RHEL 8 y OpenShift 4, Docker no está incluido y esto no funciona.

Para proporcionar compatibilidad con entornos que no tienen Docker disponible, el comando `s2i build` ahora incluye la opción `--as-dockerfile path/to/Dockerfile`. Esta opción configura el comando `s2i build` para generar un Dockerfile y dos directorios auxiliares que puede usar para compilar una imagen de contenedor de prueba a partir de un repositorio de origen y una imagen de compilador con el comando `podman build`. Mediante esta opción, no se requiere ningún daemon Docker local para ejecutar el comando `s2i build`.

Puede proporcionar la ubicación de un directorio o la URL de un repositorio Git que contiene la fuente de la aplicación como la opción `src`.

Para compilaciones incrementales, asegúrese de crear un script `save-artifacts` y enviar el indicador `--incremental` al comando `s2i build`. Si existe un script `save-artifacts`, y ya existe una imagen anterior, y usa la opción `--incremental=true`, el flujo de trabajo es el siguiente:

1. S2I crea una nueva imagen de contenedor a partir de la imagen de compilación anterior.
2. S2I ejecuta el script `save-artifacts` en este contenedor. Este script es responsable de transmitir un archivo tar de los artefactos a stdout.
3. S2I compila la nueva imagen de salida:
 - a. Los artefactos de la compilación anterior se encuentran en el directorio `artifacts` del archivo tar que se envió a la compilación.
 - b. El script `assemble` de la imagen de compilador S2I es responsable de detectar y usar los artefactos de compilación.

Ejecute los comandos `s2i --help` y `s2i subcommand --help` para obtener información sobre los diversos subcomandos, sus opciones y ejemplos sobre cómo usarlos.

Después de que haya probado la imagen de contenedor de aplicaciones localmente, puede copiar la imagen de compilador S2I a un registro para su consumo. Antes de implementar aplicaciones en un clúster OpenShift con la imagen del compilador, cree un flujo de imágenes basado en la imagen de contenedor con el comando `oc import-image`. A continuación, puede usar el flujo de imágenes para crear aplicaciones en OpenShift.

Compilación de una imagen de compilador S2I Nginx

Para crear una imagen de compilador S2I Nginx basada en RHEL 7, realice los siguientes pasos:

Crear y completar un proyecto de Dockerfile

Use el comando `s2i` para crear el directorio del proyecto de la imagen de compilador S2I y edite los archivos según sea necesario:

- Ejecute el comando `s2i create` para crear la estructura de directorio esqueleto para los artefactos de la imagen de compilador S2I:

```
[user@host ~]$ s2i create s2i-do288-nginx s2i-do288-nginx
```

- Edite el Dockerfile para incluir instrucciones para instalar los scripts S2I y Nginx. El siguiente Dockerfile para una imagen de compilador S2I Nginx usa la imagen base universal de RHEL 8:

```
FROM registry.access.redhat.com/ubi8/ubi:8.0 1

ENV X_SCLS="rh-nginx18" \
    PATH="/opt/rh/rh-nginx18/root/usr/sbin:$PATH" \
    NGINX_DOCROOT="/opt/rh/rh-nginx18/root/usr/share/nginx/html"

LABEL io.k8s.description="A Nginx S2I builder image" \
      io.k8s.display-name="Nginx 1.8 S2I builder image for DO288" \
      io.openshift.expose-services="8080:http" \
      io.openshift.s2i.scripts-url="image:///usr/libexec/s2i" \
      io.openshift.tags="builder,webserver,nginx,nginx18,html"

ADD nginxconf.sed /tmp/
COPY ./s2i/bin/ /usr/libexec/s2i 3

RUN yum install -y --nodos rh-nginx18 \
    && yum clean all \
    && sed -i -f /tmp/nginxconf.sed /etc/opt/rh/rh-nginx18/nginx/nginx.conf \
    && chgrp -R 0 /var/opt/rh/rh-nginx18 /opt/rh/rh-nginx18 5 \
    && chmod -R g=u /var/opt/rh/rh-nginx18 /opt/rh/rh-nginx18 6 \
    && echo 'Hello from the Nginx S2I builder image' > ${NGINX_DOCROOT}/index.html

EXPOSE 8080

USER 1001

CMD ["/usr/libexec/s2i/usage"]
```

- 1** Use la imagen base universal de RHEL 8 como base para la imagen de compilador S2I.
 - 2** Etiquete los metadatos para los consumidores de la imagen de compilador S2I.
 - 3** Copie los scripts S2I en la ubicación indicada por la etiqueta `io.openshift.s2i.scripts-url`.
 - 4** Instale Nginx.
 - 5** **6** Defina los permisos para ejecutar como un ID de usuario aleatorio en un clúster OpenShift.
- Cree un script `assemble` en el directorio `.s2i/bin` de la fuente de la aplicación con el siguiente contenido, que copia archivos fuente HTML a la raíz del documento del servidor web de Nginx:
- ```
#!/bin/bash -e

echo "---> Copying source HTML files to web server root..."
cp -Rf /tmp/src/. /opt/rh/rh-nginx18/root/usr/share/nginx/html/
```
- Cree un script `run` en el directorio `.s2i/bin` de la fuente de la aplicación con el siguiente contenido, que ejecuta el servidor web de Nginx en primer plano:

```
#!/bin/bash -e
exec nginx -g "daemon off;"
```

## Compilación y prueba de la imagen de compilador S2I

Use Podman y el comando `s2i` para compilar la imagen de compilador S2I y una imagen de contenedor de aplicaciones de prueba:

- Compile la imagen de compilador S2I:

```
[user@host ~]$ sudo podman build -t s2i-do288-nginx .
```

- Ejecute el comando `s2i build` para compilar una imagen de contenedor de aplicaciones de prueba. Para reemplazar el archivo predeterminado `index.html` incluido en la imagen de compilador, cree un archivo `index.html` en el directorio `test/test-app` para insertar este nuevo archivo en el contenedor Nginx de prueba:

```
[user@host ~]$ s2i build test/test-app s2i-do288-nginx nginx-test \
> --as-docker-file /path/to/Dockerfile
```

- Para compilar el contenedor de prueba, use el comando `podman build`, esta vez con el Dockerfile generado por el comando `s2i build`:

```
[user@host ~]$ sudo podman build -t nginx-test /path/to/Dockerfile
```

- Para probar la imagen de contenedor, use el comando `podman run` con un ID de usuario diferente al proporcionado en el Dockerfile, para asegurarse de que el contenedor se pueda ejecutar con un ID de usuario generado aleatoriamente en un clúster OpenShift:

```
[user@host ~]$ sudo podman run -u 1234 -d -p 8080:8080 nginx-test
```

Cuando que el contenedor se esté ejecutando sin errores, verifique que el archivo de prueba `index.html` que proporcionó en el directorio `test` se ejecute cuando pruebe el contenedor de Nginx.

## Puesta a disposición de la imagen del compilador S2I en OpenShift

Use el comando `skopeo` para enviar la imagen de compilador S2I a un registro de contenedor y crear un flujo de imágenes en OpenShift que apunte a esa imagen.

- Después de probar el contenedor localmente, envíe la imagen de compilador S2I a un registro empresarial. Por ejemplo, supongamos que tiene una cuenta de Quay.io y ha iniciado sesión con el comando `podman login`:

```
[user@host ~]$ sudo skopeo copy containers-storage:localhost/s2i-do288-nginx \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-nginx
```

- Para crear un flujo de imágenes para la imagen de compilador S2I Nginx, cree un nuevo proyecto y ejecute el comando `oc import-image`:

```
[user@host ~]$ oc import-image s2i-do288-nginx \
> --from quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-nginx \
> --confirm
```



### nota

Recuerde del Capítulo 3, *Publicación de imágenes de contenedor empresariales* que es posible que deba crear un secreto de extracción que contenga sus credenciales para el registro remoto donde publica la imagen del compilador S2I, si esa imagen no está disponible públicamente. También debe vincular ese secreto a la cuenta de servicio predeterminada usada para extraer imágenes para crear el flujo de imágenes mediante el comando `oc import-image`.

- Una vez que cree el flujo de imágenes, puede usarlo para crear aplicaciones con la imagen de compilador S2I Nginx. Tenga en cuenta que debe utilizar el tilde (~), a menos que su imagen de compilador S2I sea una alternativa de los lenguajes admitidos por el comando `oc new-app` de OpenShift:

```
[user@host ~]$ oc new-app --as-deployment-config --name nginx-test \
> s2i-do288-nginx~git_repository
```



### Referencias

Puede obtener más información en el capítulo *Creación de imágenes* de la guía *Imágenes de Red Hat OpenShift Container Platform 4.5*; en [https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.5/html-single/images/creating-images#creating-images](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/images/creating-images#creating-images)

#### Cómo crear una imagen de compilador S2I

<https://blog.openshift.com/create-s2i-builder-image>

#### Herramienta de Fuente a imagen (Source-to-Image, S2I)

<https://github.com/openshift/source-to-image>

#### Referencia del subcomando de la herramienta S2I

<https://github.com/openshift/source-to-image/blob/master/docs/cli.md>

## ► Ejercicio Guiado

# Creación de una imagen de compilador S2I

En este ejercicio, compilará y probará una imagen de compilador S2I del servidor HTTP de Apache y, luego, compilará e implementará una aplicación mediante la imagen.

## Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Compilar una imagen de compilador S2I del servidor HTTP de Apache con la herramienta s2i.
- Probar la imagen de compilador S2I localmente con una aplicación simple.
- Publicar la imagen de contenedor en el registro de contenedor de Quay.io.
- Implementar y probar una aplicación en un clúster OpenShift mediante el uso de la imagen de compilador S2I.

## Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen principal (ubi8/ubi) para la aplicación de muestra.
- La aplicación de muestra en el repositorio Git (html-helloworld).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos del ejercicio y descargar los archivos de trabajos de laboratorio y soluciones:

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

- 1. En la máquina virtual `workstation`, verifique que el paquete `source-to-image` esté instalado, que proporciona la herramienta de línea de comandos `s2i`:

```
[student@workstation ~]$ s2i version
s2i v1.1.13
```

- 2. Use el comando `s2i` para crear los archivos de plantillas y directorios necesarios para la imagen de compilador S2I.
- 2.1. En la máquina virtual `workstation`, use el comando `s2i create` para crear los archivos de plantillas para la imagen del compilador en el directorio `/home/student/`:

```
[student@workstation ~]$ s2i create s2i-do288-httpd s2i-do288-httpd
```

- 2.2. Verifique que se hayan creado los archivos de plantillas. El comando `s2i create` crea la siguiente estructura de directorio:

```
[student@workstation ~]$ tree -a s2i-do288-httdp
s2i-do288-httdp
├── Dockerfile
├── Makefile
└── README.md
└── s2i
 ├── bin
 │ ├── assemble
 │ ├── run
 │ ├── save-artifacts
 │ └── usage
 └── test
 ├── run
 └── test-app
 └── index.html
4 directories, 9 files
```

► 3. Cree la imagen de compilador S2I del servidor HTTP de Apache.

- 3.1. Se proporciona un ejemplo de Dockerfile para la imagen de compilador del servidor HTTP de Apache en `~/D0288/labs/apache-s2i/Dockerfile`. Revise brevemente este archivo:

```
[student@workstation ~]$ cat ~/D0288/labs/apache-s2i/Dockerfile
FROM registry.access.redhat.com/ubi8/ubi:8.0 ①

Generic labels
LABEL Component="httpd" \
 Name="s2i-do288-httdp" \
 Version="1.0" \
 Release="1"

Labels consumed by OpenShift
LABEL io.k8s.description="A basic Apache HTTP Server S2I builder image" \
 io.k8s.display-name="Apache HTTP Server S2I builder image for D0288" \
 io.openshift.expose-services="8080:http" \
 io.openshift.s2i.scripts-url="image:///usr/libexec/s2i" ③

This label is used to categorize this image as a builder image in the
OpenShift web console.
LABEL io.openshift.tags="builder, httpd, httpd24"

Apache HTTP Server DocRoot
ENV DOCROOT /var/www/html

RUN yum install -y --nодocs --disableplugin=subscription-manager httpd && \
 yum clean all --disableplugin=subscription-manager -y && \
 echo "This is the default index page from the s2i-do288-httdp S2I builder \
 image." > ${DOCROOT}/index.html ⑤

Change web server port to 8080
```

```
RUN sed -i "s/Listen 80/Listen 8080/g" /etc/httpd/conf/httpd.conf

Copy the S2I scripts to the default location indicated by the
io.openshift.s2i.scripts-url LABEL (default is /usr/libexec/s2i)
COPY ./s2i/bin/ /usr/libexec/s2i 6
...output omitted...
```

- 1** Use la imagen base universal de RHEL 8 como imagen base para este contenedor.
- 2** Establezca las etiquetas que se usan para OpenShift para describir la imagen de compilador.
- 3** Configure dónde están ubicados los scripts S2I obligatorios (`run`, `assemble`)
- 4** Instale el paquete del servidor web `httpd` y limpie la memoria caché de Yum.
- 5** Establezca el contenido del archivo predeterminado `index.html` para la imagen de compilador.
- 6** Copie los scripts S2I en el directorio `/usr/libexec/s2i`.

A continuación, copie este Dockerfile en el directorio `~/s2i-do288-httpd` y sobrescriba el Dockerfile de plantilla generado:

```
[student@workstation ~]$ cp ~/D0288/labs/apache-s2i/Dockerfile ~/s2i-do288-httpd/
```

- 3.2. De manera similar, revise y copie los scripts S2I para la imagen de compilador proporcionados en el directorio `~/D0288/labs/apache-s2i/s2i/bin` en el directorio `~/s2i-do288-httpd/s2i/bin` y sobrescriba los scripts generados:

```
[student@workstation ~]$ cp -Rv ~/D0288/labs/apache-s2i/s2i ~/s2i-do288-httpd/
```

- 3.3. Este ejercicio no implementa el script `save-artifacts`. Elimínelo del directorio `~/s2i-do288-httpd/s2i/bin`:

```
[student@workstation ~]$ rm -f ~/s2i-do288-httpd/s2i/bin/save-artifacts
```

- 3.4. Cree la imagen de compilador S2I del servidor HTTP de Apache. No se olvide del punto al final del comando `podman build`. Indica que Docker debe usar el Dockerfile en el directorio actual para compilar la imagen:

```
[student@workstation ~]$ cd s2i-do288-httpd
[student@workstation s2i-do288-httpd]$ sudo podman build -t s2i-do288-httpd .
STEP 1: FROM registry.access.redhat.com/ubi8/ubi:8.0
...output omitted...
STEP 25: COMMIT s2i-do288-httpd
```

**nota**

Puede omitir de forma segura el siguiente error en la salida de compilación:  
**ERRO[0000] HOSTNAME is not supported for OCI image format,**  
**hostname 1d561c58fd2b will be ignored. Must use `docker`**  
**format** (Nombre de host no soportado para el formato de imagen OCI; el nombre  
de host 1d561c58fd2b se omitirá. Debe usar formato "docker"). Este es un error en  
la versión de la herramienta Podman y es un mensaje de error falso; la compilación  
del contenedor continúa sin problema a pesar del error. Puede encontrar más  
información en este Bugzilla [[https://bugzilla.redhat.com/show\\_bug.cgi?id=1661592](https://bugzilla.redhat.com/show_bug.cgi?id=1661592)]

- 3.5. Verifique que se haya creado la imagen de compilador:

```
[student@workstation s2i-do288-httdp]$ sudo podman images
REPOSITORY TAG IMAGE ID CREATED
localhost/s2i-do288-httdp latest 82beb27428b7 9 seconds ago
registry.access.redhat.com/ubi8/ubi 8.0 7ae69d957d8b 2 weeks ago
...output omitted...
```

- ▶ 4. Compile y pruebe la imagen de contenedor de aplicaciones que combina la imagen de compilador S2I del servidor HTTP de Apache y el código fuente de la aplicación.
- 4.1. Cuando la imagen del compilador esté lista, puede usar el comando `s2i build` para compilar la imagen de contenedor de aplicaciones. Antes de compilar la imagen del contenedor de aplicaciones, revise el archivo HTML de muestra `~/D0288/labs/apache-s2i/index.html`:

```
[student@workstation s2i-do288-httdp]$ cat ~/D0288/labs/apache-s2i/index.html
This is the index page from the app
```

A continuación, copie este archivo en el directorio `~/s2i-do288-httdp/test/test-app` para reemplazar el archivo `index.html` empaquetado dentro de la imagen de compilador:

```
[student@workstation s2i-do288-httdp]$ cp ~/D0288/labs/apache-s2i/index.html \
> ~/s2i-do288-httdp/test/test-app/
```

- 4.2. Cree un nuevo directorio para el Dockerfile generado por el comando `s2i build`.

```
[student@workstation s2i-do288-httdp]$ mkdir /home/student/s2i-sample-app
```

- 4.3. Compile la imagen del contenedor de aplicaciones:

```
[student@workstation s2i-do288-httdp]$ s2i build test/test-app/ \
> s2i-do288-httdp s2i-sample-app \
> --as-dockerfile ~/s2i-sample-app/Dockerfile
Application dockerfile generated in /home/student/s2i-sample-app/Dockerfile
```

- 4.4. Revise el directorio de aplicaciones generado.

```
[student@workstation s2i-do288-httdp]$ cd ~/s2i-sample-app
[student@workstation s2i-sample-app]$ tree .
.
├── Dockerfile
├── downloads
│ ├── defaultScripts
│ └── scripts
└── upload
 ├── scripts
 └── src
 └── index.html

6 directories, 2 files
```

Observe el archivo `index.html` ubicado en el directorio `upload`. Este es el mismo archivo `index.html` que copió en el directorio `test/test-app` en un paso anterior.

#### 4.5. Revise el Dockerfile generado.

```
[student@workstation s2i-sample-app]$ cat Dockerfile
FROM s2i-do288-httdp ①
LABEL "io.k8s.display-name"="s2i-sample-app" \
 "io.openshift.s2i.build.image"="s2i-do288-httdp" \
 "io.openshift.s2i.build.source-location"="test/test-app/"

USER root
Copying in source code
COPY upload/src /tmp/src ③
Change file ownership to the assemble user. Builder image must support chown
command.
RUN chown -R 1001:0 /tmp/src
USER 1001
Assemble script sourced from builder image based on user input or image
metadata.
If this file does not exist in the image, the build will fail.
RUN /usr/libexec/s2i/assemble
Run script sourced from builder image based on user input or image metadata.
If this file does not exist in the image, the build will fail.
CMD /usr/libexec/s2i/run
```

- ① Use la imagen de compilador `s2i-do288-httdp` como elemento principal de esta imagen de contenedor.
- ② Establezca las etiquetas que se usan para OpenShift para describir la aplicación.
- ③ Copie en el código fuente contenido en el directorio `upload/src`.

#### 4.6. Compile una imagen de contenedor de prueba desde el Dockerfile generado.

```
[student@workstation s2i-sample-app]$ sudo podman build \
> --format docker -t s2i-sample-app .
STEP 1: FROM s2i-do288-httdp
STEP 2: LABEL "io.k8s.display-name"="s2i-sample-app"...output omitted...
...output omitted...
STEP 15: COMMIT s2i-sample-app
```

- 4.7. Verifique que se haya creado la imagen del contenedor de aplicaciones:

```
[student@workstation s2i-sample-app]$ sudo podman images
REPOSITORY TAG IMAGE ID CREATED
localhost/s2i-sample-app latest 3c8637c4372d About an hour ago
localhost/s2i-do288-httdp latest d06040a9eeaca About an hour ago
...output omitted...
```

- 4.8. Pruebe la imagen del contenedor de aplicaciones localmente en la máquina virtual `workstation`. Ejecute el contenedor como un usuario aleatorio con el indicador `-u` para simular la ejecución como un usuario aleatorio en un clúster OpenShift. Puede copiar el comando o ejecutarlo directamente del script `~/DO288/labs/apache-s2i/local-test.sh`:

```
[student@workstation s2i-sample-app]$ sudo podman run --name test -u 1234 \
> -p 8080:8080 -d s2i-sample-app
a5f4b3e5bf...output omitted...
```

- 4.9. Verifique que el contenedor se inició correctamente:

```
[student@workstation s2i-sample-app]$ sudo podman ps
CONTAINER ID IMAGE COMMAND ...
a5f4b3e5bfaa localhost/s2i-sample-app:latest /bin/sh -c /usr/lib...
...output omitted...
```

- 4.10. Use el comando `curl` para probar la aplicación:

```
[student@workstation s2i-sample-app]$ curl http://localhost:8080
This is the index page from the app
```

- 4.11. Detenga el contenedor de aplicaciones de prueba:

```
[student@workstation s2i-sample-app]$ sudo podman stop test
a5f4b3e5bf...output omitted...
```

► 5. Envíe la imagen de compilador S2I del servidor HTTP de Apache en su cuenta de Quay.io.

- 5.1. 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 s2i-sample-app]$ source /usr/local/etc/ocp4.config
```

- 5.2. Inicie sesión en su cuenta de Quay.io con el comando `podman`. Se le solicitará que ingrese su contraseña.

```
[student@workstation s2i-sample-app]$ sudo podman login \
> -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 5.3. Use el comando `skopeo copy` para publicar la imagen de compilador S2I en su cuenta de Quay.io.

```
[student@workstation s2i-sample-app]$ sudo skopeo copy \
> containers-storage:localhost/s2i-do288-httd \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-httd \
...output omitted...
Writing manifest to image destination
Storing signatures
```

► 6. Cree un flujo de imágenes para la imagen de compilador S2I del servidor HTTP de Apache.

- 6.1. Inicie sesión en OpenShift y cree un proyecto nuevo. Coloque como prefijo del nombre del proyecto su nombre de usuario de desarrollador.

```
[student@workstation s2i-sample-app]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation s2i-sample-app]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-apache-s2i
Now using project "youruser-apache-s2i" on server "https://
api.cluster.domain.example.com:6443".
```

- 6.2. Inicie sesión en su cuenta personal de Quay.io con Podman, esta vez sin el comando `sudo`, para poder exportar el archivo `auth.json` y usarlo en un secreto de extracción en el siguiente paso. Debe ingresar su contraseña nuevamente.

```
[student@workstation s2i-sample-app]$ podman login \
> -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 6.3. Cree un secreto con el token de acceso de la API de registro de contenedores almacenado por Podman.

También puede ejecutar o cortar y pegar el comando `oc create secret` siguiente del script `create-secret.sh` en el directorio `/home/student/D0288/labs/apache-s2i`.

```
[student@workstation s2i-sample-app]$ oc create secret generic quayio \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type=kubernetes.io/dockerconfigjson
secret/quayio created
```

- 6.4. Vincule el secreto nuevo con la cuenta de servicio `builder`.

```
[student@workstation s2i-sample-app]$ oc secrets link builder quayio
```

- 6.5. Cree un flujo de imágenes al importar la imagen de compilador S2I del registro del contenedor de Quay.io:

```
[student@workstation s2i-sample-app]$ oc import-image s2i-do288-httd \
> --from quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-httd --confirm
imagestream.image.openshift.io/s2i-do288-httd imported

Name: s2i-do288-httd
Namespace: youruser-apache-s2i
Created: Less than a second ago
Labels: <none>
Annotations: openshift.io/image.dockerRepositoryCheck=2019-06-26T02:30:59Z
Image Repository: image-registry.openshift-image-registry.svc:5000/youruser-
apache-s2i/s2i-do288-httd
Image Lookup: local=false
Unique Images: 1
Tags: 1

latest
tagged from quay.io/youruser/s2i-do288-httd

* quay.io/youruser/s2i-do288-httd@sha256:fe0cd09432...
 Less than a second ago

...output omitted...
```

6.6. Verifique que se haya creado el flujo de imágenes:

```
[student@workstation s2i-sample-app]$ oc get is
NAME IMAGE REPOSITORY ...output omitted...
s2i-do288-httd ...youruser-apache-s2i/s2i-do288-httd
```

- 7. Implemente y pruebe la aplicación html-helloworld del repositorio Git del aula en un clúster OpenShift. Esta aplicación consta de un único archivo HTML que muestra un mensaje.
- 7.1. Cree una nueva aplicación denominada hello a partir de fuentes en Git. Necesita anteponer la URL de Git con el flujo de imágenes s2i-do288-httd para asegurarse de que la aplicación use la imagen de compilador del servidor HTTP de Apache que creó antes:

```
[student@workstation s2i-sample-app]$ oc new-app --as-deployment-config \
> --name hello-s2i \
> s2i-do288-httd~https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
> --context-dir=html-helloworld
--> Found image c7a496d (2 hours old) in image stream "youruser-apache-s2i/s2i-
do288-httd" under tag "latest" for "s2i-do288-httd"

Apache HTTP Server S2I builder image for D0288

A basic Apache HTTP Server S2I builder image
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
...output omitted...
```

**capítulo 5 |** Personalización de compilaciones de fuente a imagen

- 7.2. Vea los registros de compilación. Espere hasta que termine la compilación y la imagen del contenedor de aplicaciones se envíe al registro OpenShift:

```
[student@workstation s2i-sample-app]$ oc logs -f bc/hello-s2i
Cloning "https://github.com/youruser/D0288-apps" ...
...output omitted...
---> Copying source files to web server directory...
Pushing image-registry.openshift-image-registry.svc:5000/youruser-apache-s2i/
hello-s2i:latest ...
...output omitted...
Push successful
```

- 7.3. Espere hasta que la aplicación se haya implementado. Consulte el estado del pod de la aplicación. El pod de la aplicación debe estar en el estado **Running** (En ejecución):

```
[student@workstation s2i-sample-app]$ oc get pods
NAME READY STATUS RESTARTS AGE
hello-s2i-1-build 0/1 Completed 0 2m
hello-s2i-1-deploy 0/1 Completed 0 72s
hello-s2i-1-w8nqp 1/1 Running 0 63s
```

- 7.4. Exponga la aplicación a un acceso externo usando una ruta:

```
[student@workstation s2i-sample-app]$ oc expose svc hello-s2i
route.route.openshift.io/hello-s2i exposed
```

- 7.5. Obtenga la URL de enrutamiento con el comando `oc get route`:

```
[student@workstation s2i-sample-app]$ oc get route/hello-s2i \
> -o jsonpath='{.spec.host}{"\n"}'
hello-s2i-youruser-apache-s2i.apps.cluster.domain.example.com
```

- 7.6. Pruebe la aplicación con la URL de ruta que obtuvo en el paso anterior:

```
[student@workstation s2i-sample-app]$ curl \
> http://hello-s2i-${RHT_OCP4_DEV_USER}-apache-s2i.${RHT_OCP4_WILDCARD_DOMAIN}
<html>
<body>
 <h1>Hello, World!</h1>
</body>
</html>
```

► **8.** Realice la limpieza.

- 8.1. Elimine el proyecto `apache-s2i` en OpenShift:

```
[student@workstation s2i-sample-app]$ oc delete project \
> ${RHT_OCP4_DEV_USER}-apache-s2i
```

- 8.2. Elimine el contenedor `test` que creó antes cuando probó la aplicación localmente:

```
[student@workstation s2i-sample-app]$ sudo podman rm test
a5f4b3e5bf...output omitted...
```

- 8.3. Elimine las imágenes de contenedor de la máquina virtual **workstation**:

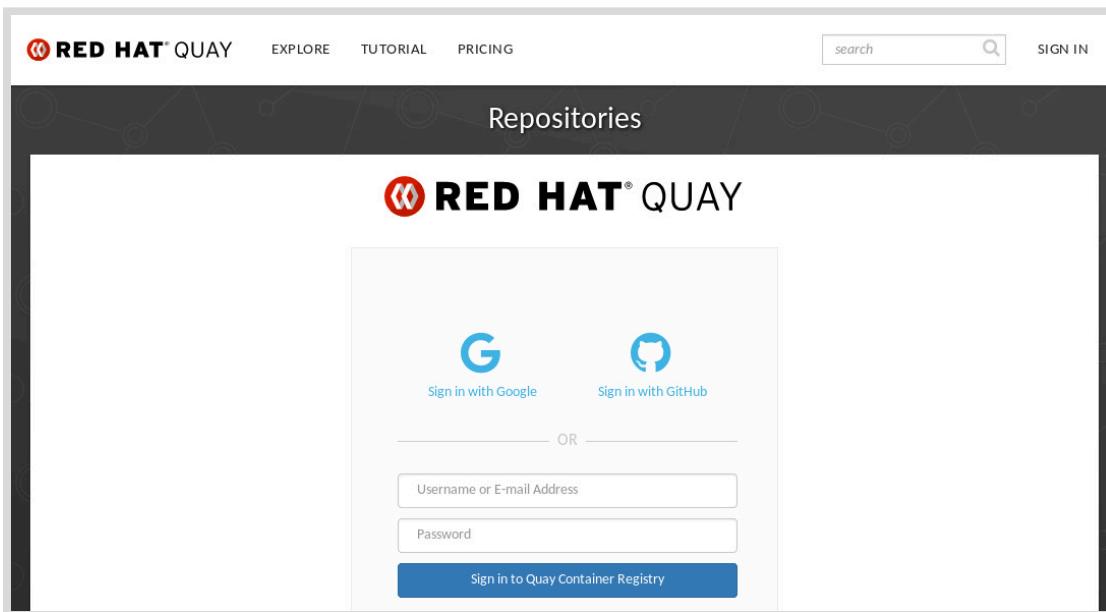
```
[student@workstation s2i-sample-app]$ sudo podman rmi -f \
> localhost/s2i-sample-app \
> localhost/s2i-do288-httdp \
> registry.access.redhat.com/ubi8/ubi:8.0
...output omitted...
```

- 8.4. Elimine la imagen **s2i-do288-httdp** del registro externo:

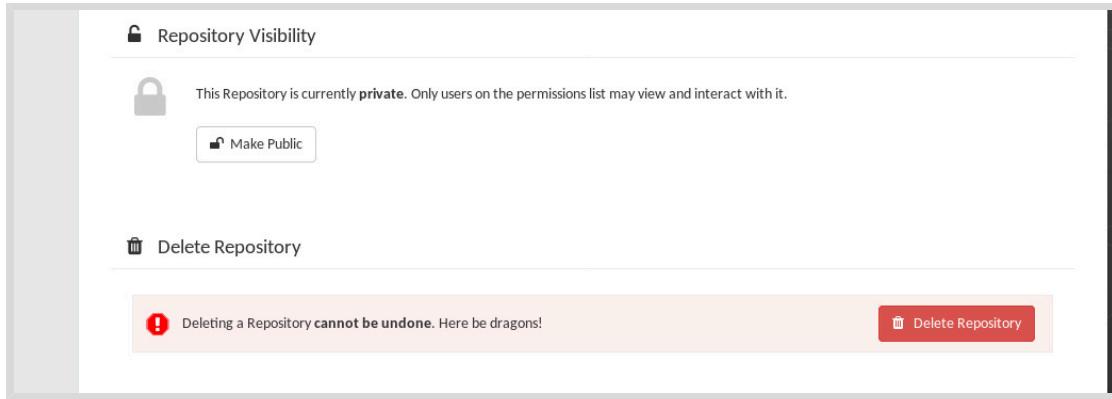
```
[student@workstation s2i-sample-app]$ sudo skopeo delete \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-httdp:latest
```

- 8.5. Inicie sesión en Quay.io con su cuenta personal gratuita.

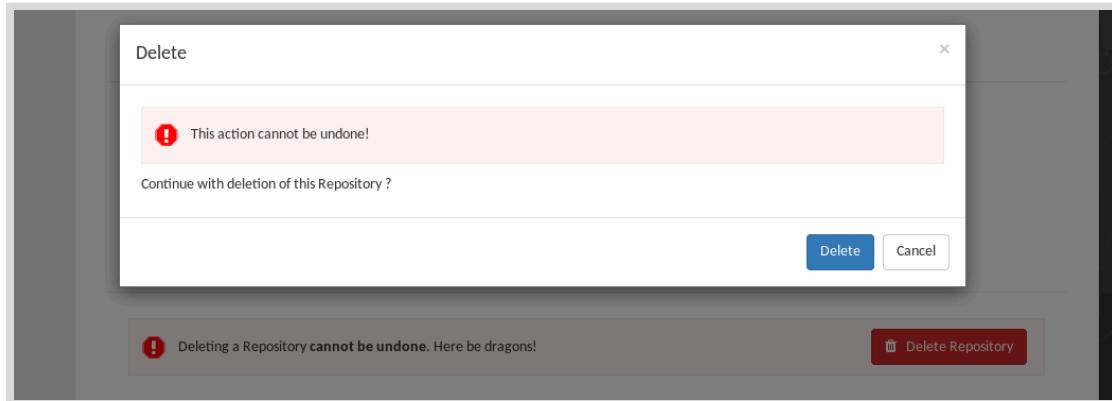
Desplácese hasta <http://quay.io> y haga clic en **Sign In** (Iniciar sesión) para proporcionar sus credenciales de usuario. Haga clic en **Sign in to Quay Container Registry** (Iniciar sesión en el registro de contenedor de Quay) para iniciar sesión en Quay.io.



- 8.6. En el menú principal de Quay.io, haga clic en **Repositories** (Repositorios) y busque **s2i-do288-httdp**. El ícono de candado que está al lado indica que es un repositorio privado que requiere autenticación para extracciones e inserciones. Haga clic en **s2i-do288-httdp** para mostrar la página **Repository Activity** (Actividad del repositorio).
- 8.7. En la página **Repository Activity** (Actividad del repositorio) del repositorio **s2i-do288-httdp**, desplácese hacia abajo y haga clic en el ícono de engranaje para mostrar la pestaña **Settings** (Configuración). Desplácese hacia abajo y haga clic en **Delete Repository** (Eliminar repositorio).



- 8.8. En el cuadro de diálogo **Delete** (Eliminar), haga clic en **Delete** (Eliminar) para confirmar que desea eliminar el repositorio **s2i-do288-[httpd](#)**. Después de unos momentos, volverá a la página **Repositories** (Repositorios). Ahora puede cerrar sesión en Quay.io.



## Finalizar

En la máquina virtual **workstation**, ejecute el script **lab apache-s2i finish** para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab apache-s2i finish
```

Esto concluye el ejercicio guiado.

## ► Trabajo de laboratorio

# Personalización de compilaciones de fuente a imagen

### Listado de verificación de rendimiento

En este trabajo de laboratorio, creará una imagen de compilador S2I para aplicaciones en ejecución basado en el lenguaje de programación Go.

### Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Compilar una imagen de compilador S2I del lenguaje de programación Go basada en RHEL 8.
- Probar la imagen de compilador S2I localmente mediante la creación de un Dockerfile con la herramienta s2i y, luego, la compilación y ejecución de la imagen de contenedor resultante con Podman.
- Publicar la imagen de compilador en su cuenta personal Quay.io.
- Usar la imagen de compilador S2I para implementar y probar una aplicación en un clúster OpenShift.
- Personalizar el script run para cambiar el comportamiento de la aplicación y volver a implementar la aplicación para probar los cambios.

### Antes De Comenzar

Para realizar este trabajo de laboratorio, debe tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La herramienta de línea de comandos s2i.
- Una bifurcación del repositorio Git que contiene el código fuente de la aplicación (go-hello).

Ejecute el siguiente comando en la máquina virtual workstation para validar los requisitos previos. Este comando también descarga archivos auxiliares y de solución para el trabajo de laboratorio de revisión:

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

El script de configuración crea una carpeta denominada `custom-s2i` en la carpeta `/home/student/D0288/labs`. Esta carpeta contiene la estructura de la carpeta de la plantilla y los archivos creados por el comando `s2i create`.

### Requisitos

En este trabajo de laboratorio, debe implementar una aplicación que esté escrita en el lenguaje de programación Go.

Se proporciona un pequeño ejemplo de aplicación Go para que use para probar la imagen de compilador S2I. El ejemplo de aplicación Go proporciona una API HTTP simple que responde a solicitudes según el recurso solicitado en la solicitud HTTP.

Para completar este trabajo de laboratorio, compile una imagen de compilador S2I para aplicaciones basadas en Go, y pruebe e implemente un ejemplo de aplicación Go en un clúster OpenShift según los siguientes requisitos:

- La imagen del compilador S2I debe llamarse `s2i-do288-go`. La imagen de compilador debe estar disponible desde su cuenta de Quay.io personal en:

`quay.io/youruser/s2i-do288-go`

- El flujo de imágenes para la imagen de compilador S2I debe llamarse `s2i-do288-go`.
- El nombre de la aplicación para OpenShift es `greet`.

• Tanto el flujo de imágenes como la aplicación deben crearse dentro de un proyecto denominado `youruser-custom-s2i`.

- Se debe poder acceder a la API HTTP para la aplicación en la siguiente URL:

`http://greet-youruser-custom-s2i.apps.cluster.domain.example.com`.

- El código fuente de la aplicación Go se encuentra dentro del repositorio Git `D0288-apps` en el directorio `go-hello`.

- Antes de enviar la imagen de compilador en su cuenta de Quay.io, pruebe la aplicación localmente en la máquina virtual `workstation`. Tenga en cuenta lo siguiente cuando pruebe la imagen de compilador:

- El código fuente de la aplicación está ubicado en la carpeta `~/D0288/labs/custom-s2i/test/test-app`.

- Asignele el nombre `s2i-go-app` a la imagen del contenedor de aplicaciones de prueba.

- Asignele el nombre `go-test` al contenedor de prueba.

- Asegúrese de que cuando realice una prueba del contenedor, use un ID de usuario aleatorio, como `1234`, para simular la ejecución en un clúster OpenShift.

- Vincule el puerto del contenedor `8080` con el puerto local `8080`.

- La aplicación devuelve un saludo basado en la URL que realizó la solicitud. Por ejemplo:

- `http://localhost:8080/user1`, devuelve la siguiente respuesta:

```
Hello user1!. Welcome!
```

- Cuando finalice las pruebas, elimine el contenedor `go-test` antes de ir al siguiente paso.

- Pruebe la compilación de la aplicación `go-hello` desde el origen con la imagen de compilador `s2i-do288-go`.

- Después de probar la aplicación `go-hello` que se ejecuta en OpenShift, personalice el script `run` para la imagen de compilador `s2i-do288-go`, mediante su reemplazo en el código fuente de la aplicación `go-hello`.

- Confirme e inserte el script `run` personalizado en el repositorio de Github usado como origen para la compilación de la aplicación. A continuación, inicie una nueva compilación y verifique que la nueva versión de la aplicación `go-hello` devuelva el saludo en español, como:

```
Hola user1!. Bienvenido!
```

## Pasos

1. Los scripts S2I para la imagen de compilador se proporcionaron en la carpeta `/home/student/D0288/labs/custom-s2i/s2i/bin`. Revise los scripts `assemble`, `run` y `usage`.
2. Edite el Dockerfile para la imagen de compilador para incluir una instrucción para copiar los scripts S2I en la ubicación adecuada de la imagen de compilador. Agregue esta nueva instrucción inmediatamente después del comentario `TODO` ya presente en el archivo.
3. Compile la imagen del compilador S2I. Asígnale el nombre `s2i-do288-go` a la imagen.
4. Compile un Dockerfile para una imagen del contenedor de aplicaciones que combina la imagen de compilador S2I y el código fuente de la aplicación localmente en la máquina virtual `workstation` en el directorio `/home/student/D0288/labs/custom-s2i/test/test-app`.
5. Envíe la imagen de compilador S2I `s2i-do288-go` a su cuenta personal de Quay.io.
6. Cree un flujo de imágenes denominado `s2i-do288-go` para la imagen de compilador S2I `s2i-do288-go`. Cree el flujo de imágenes en un proyecto denominado `youruser-custom-s2i`.
7. Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación `master` del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:
8. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio y envíela a Github:
9. Implemente y pruebe la aplicación `go-hello` desde su bifurcación personal de Github del repositorio `D0288-apps` al clúster OpenShift del aula. Asegúrese de hacer referencia a la bifurcación `custom-s2i` que creó en el paso anterior al implementar la aplicación. La aplicación devuelve un saludo al recurso solicitado por la solicitud HTTP. Por ejemplo, invocar la aplicación con la siguiente URL:  
`http://greet-youruser-custom-s2i.apps.cluster.domain.example.com/user1`, devuelve la siguiente respuesta:

```
Hello user1!. Welcome!
```

10. Personalice el script `run` para la imagen de compilador `s2i-do288-go` en la fuente de la aplicación. Cambie la manera en que se inicia la aplicación; para ello, agregue un argumento `--lang es` en el inicio. Esto cambia el idioma predeterminado que usa la aplicación. Confirme y envíe los cambios a la bifurcación que usó como fuente de entrada para la compilación de la aplicación.

11. Vuelva a compilar y pruebe la aplicación. La aplicación ahora debe responder a solicitudes en español. Por ejemplo, invocar la aplicación con la siguiente URL:

`http://greet-youruser-custom-s2i.apps.cluster.domain.example.com/user1`, devuelve la siguiente respuesta:

```
Hola user1!. Bienvenido!
```

12. Califique su trabajo.

Ejecute el siguiente comando en la máquina virtual `workstation` para verificar que se hayan realizado todas las tareas:

```
[student@workstation ~]$ lab custom-s2i grade
```

13. Realice la limpieza. Realice los siguientes pasos:

- 13.1. Elimine el proyecto `youruser-custom-s2i` de OpenShift.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-custom-s2i
```

- 13.2. Elimine todos los contenedores de prueba creados antes para probar la aplicación localmente.

```
[student@workstation ~]$ sudo podman rm go-test
```

- 13.3. Elimine todas las imágenes de contenedor compiladas durante este trabajo de laboratorio en la máquina virtual `workstation`.

```
[student@workstation ~]$ sudo podman rmi -f \
> localhost/s2i-go-app \
> localhost/s2i-do288-go \
> registry.access.redhat.com/ubi8/ubi:8.0
...output omitted...
```

- 13.4. Elimine la imagen `s2i-do288-go` del registro externo:

```
[student@workstation ~]$ sudo skopeo delete \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-go:latest
```

- 13.5. Inicie sesión en Quay.io con su cuenta personal gratuita.

Desplácese hasta `http://quay.io` y haga clic en **Sign In** (Iniciar sesión) para proporcionar sus credenciales de usuario. Haga clic en **Sign in to Quay Container Registry** (Iniciar sesión en el registro de contenedor de Quay) para iniciar sesión en Quay.io.

- 13.6. En el menú principal de Quay.io, haga clic en **Repositories** (Repositorios) y busque `s2i-do288-go`. El ícono de candado indica que es un repositorio privado que requiere autenticación para extracciones e inserciones. Haga clic en `s2i-do288-go` para mostrar la página **Repository Activity** (Actividad del repositorio).

- 13.7. En la página **Repository Activity** (Actividad del repositorio) del repositorio `s2i-do288-go`, desplácese hacia abajo y haga clic en el ícono de engranaje para mostrar

la pestaña **Settings** (Configuración). Desplácese hacia abajo y haga clic en **Delete Repository** (Eliminar repositorio).

- 13.8. En el cuadro de diálogo **Delete** (Eliminar), haga clic en **Delete** (Eliminar) para confirmar que desea eliminar el repositorio `s2i-do288-go`. Después de unos momentos, volverá a la página **Repositories** (Repositorios). Ahora puede cerrar sesión en Quay.io.

## Finalizar

En la máquina virtual `workstation`, ejecute el comando `lab custom-s2i finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab custom-s2i finish
```

Esto concluye el trabajo de laboratorio.

## ► Solución

# Personalización de compilaciones de fuente a imagen

### Lista de verificación de rendimiento

En este trabajo de laboratorio, creará una imagen de compilador S2I para aplicaciones en ejecución basado en el lenguaje de programación Go.

### Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Compilar una imagen de compilador S2I del lenguaje de programación Go basada en RHEL 8.
- Probar la imagen de compilador S2I localmente mediante la creación de un Dockerfile con la herramienta s2i y, luego, la compilación y ejecución de la imagen de contenedor resultante con Podman.
- Publicar la imagen de compilador en su cuenta personal Quay.io.
- Usar la imagen de compilador S2I para implementar y probar una aplicación en un clúster OpenShift.
- Personalizar el script run para cambiar el comportamiento de la aplicación y volver a implementar la aplicación para probar los cambios.

### Andes De Comenzar

Para realizar este trabajo de laboratorio, debe tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La herramienta de línea de comandos s2i.
- Una bifurcación del repositorio Git que contiene el código fuente de la aplicación (go-hello).

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos. Este comando también descarga archivos auxiliares y de solución para el trabajo de laboratorio de revisión:

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

El script de configuración crea una carpeta denominada `custom-s2i` en la carpeta `/home/student/D0288/labs`. Esta carpeta contiene la estructura de la carpeta de la plantilla y los archivos creados por el comando `s2i create`.

### Requisitos

En este trabajo de laboratorio, debe implementar una aplicación que esté escrita en el lenguaje de programación Go.

Se proporciona un pequeño ejemplo de aplicación Go para que use para probar la imagen de compilador S2I. El ejemplo de aplicación Go proporciona una API HTTP simple que responde a solicitudes según el recurso solicitado en la solicitud HTTP.

Para completar este trabajo de laboratorio, compile una imagen de compilador S2I para aplicaciones basadas en Go, y pruebe e implemente un ejemplo de aplicación Go en un clúster OpenShift según los siguientes requisitos:

- La imagen del compilador S2I debe llamarse `s2i-do288-go`. La imagen de compilador debe estar disponible desde su cuenta de Quay.io personal en:

`quay.io/youruser/s2i-do288-go`

- El flujo de imágenes para la imagen de compilador S2I debe llamarse `s2i-do288-go`.
- El nombre de la aplicación para OpenShift es `greet`.
- Tanto el flujo de imágenes como la aplicación deben crearse dentro de un proyecto denominado `youruser-custom-s2i`.

- Se debe poder acceder a la API HTTP para la aplicación en la siguiente URL:

`http://greet-youruser-custom-s2i.apps.cluster.domain.example.com`

- El código fuente de la aplicación Go se encuentra dentro del repositorio Git `D0288-apps` en el directorio `go-hello`.
- Antes de enviar la imagen de compilador en su cuenta de Quay.io, pruebe la aplicación localmente en la máquina virtual `workstation`. Tenga en cuenta lo siguiente cuando pruebe la imagen de compilador:

- El código fuente de la aplicación está ubicado en la carpeta `~/D0288/labs/custom-s2i/test/test-app`.

- Asígnele el nombre `s2i-go-app` a la imagen del contenedor de aplicaciones de prueba.

- Asígnele el nombre `go-test` al contenedor de prueba.

- Asegúrese de que cuando realice una prueba del contenedor, use un ID de usuario aleatorio, como `1234`, para simular la ejecución en un clúster OpenShift.

- Vincule el puerto del contenedor `8080` con el puerto local `8080`.

- La aplicación devuelve un saludo basado en la URL que realizó la solicitud. Por ejemplo:

`http://localhost:8080/user1`, devuelve la siguiente respuesta:

```
Hello user1!. Welcome!
```

- Cuando finalice las pruebas, elimine el contenedor `go-test` antes de ir al siguiente paso.
- Pruebe la compilación de la aplicación `go-hello` desde el origen con la imagen de compilador `s2i-do288-go`.
- Después de probar la aplicación `go-hello` que se ejecuta en OpenShift, personalice el script `run` para la imagen de compilador `s2i-do288-go`, mediante su reemplazo en el código fuente de la aplicación `go-hello`.

- Confirme e inserte el script `run` personalizado en el repositorio de Github usado como origen para la compilación de la aplicación. A continuación, inicie una nueva compilación y verifique que la nueva versión de la aplicación go-hello devuelva el saludo en español, como:

```
Hola user1!. Bienvenido!
```

## Pasos

1. Los scripts S2I para la imagen de compilador se proporcionaron en la carpeta `/home/student/D0288/labs/custom-s2i/s2i/bin`. Revise los scripts `assemble`, `run` y `usage`.
2. Edite el Dockerfile para la imagen de compilador para incluir una instrucción para copiar los scripts S2I en la ubicación adecuada de la imagen de compilador. Agregue esta nueva instrucción inmediatamente después del comentario `TODO` ya presente en el archivo. Edite el Dockerfile en `~/D0288/labs/custom-s2i/Dockerfile` y agregue la siguiente instrucción `COPY` después del comentario `TODO`. También puede copiar la instrucción del Dockerfile de solución provisto en `~/D0288/solutions/custom-s2i/Dockerfile`:

```
COPY ./s2i/bin/ /usr/libexec/s2i
```

3. Compile la imagen del compilador S2I. Asígnale el nombre `s2i-do288-go` a la imagen.

- 3.1. Cree la imagen de compilador S2I:

```
[student@workstation ~]$ cd ~/D0288/labs/custom-s2i
[student@workstation custom-s2i]$ sudo podman build -t s2i-do288-go .
STEP 1: FROM registry.access.redhat.com/ubi8/ubi:8.0
Getting image source signatures
...output omitted...
Installed:
 golang-1.12.8-2.module+el8.1.0+4089+be929cf8.x86_64
 ...output omitted...
STEP 23: COMMIT s2i-do288-go
```



### nota

Puede omitir de forma segura el siguiente error en la salida de compilación:  
`ERRO[0000] HOSTNAME is not supported for OCI image format, hostname 1d561c58fd2b will be ignored. Must use `docker` format` (Nombre de host no soportado para el formato de imagen OCI; el nombre de host 1d561c58fd2b se omitirá. Debe usar formato "docker"). Este es un error en la versión de la herramienta Podman y es un mensaje de error falso; la compilación del contenedor continúa sin problema a pesar del error. Puede encontrar más información en este Bugzilla [[https://bugzilla.redhat.com/show\\_bug.cgi?id=1661592](https://bugzilla.redhat.com/show_bug.cgi?id=1661592)]

- 3.2. Verifique que se haya creado la imagen de compilador:

```
[student@workstation custom-s2i]$ sudo podman images
REPOSITORY TAG IMAGE ID ...
localhost/s2i-do288-go latest d1f856d10fa7 ...
...output omitted...
```

4. Compile un Dockerfile para una imagen del contenedor de aplicaciones que combina la imagen de compilador S2I y el código fuente de la aplicación localmente en la máquina virtual workstation en el directorio /home/student/D0288/labs/custom-s2i/test/test-app.
  - 4.1. Cree un directorio denominado s2i-go-app donde el comando s2i pueda guardar el Dockerfile.

```
[student@workstation custom-s2i]$ mkdir /home/student/s2i-go-app
```

- 4.2. Use el comando s2i build para generar un Dockerfile para la imagen del contenedor de aplicaciones:

```
[student@workstation custom-s2i]$ s2i build test/test-app/ \
> s2i-do288-go s2i-go-app \
> --as-dockerfile /home/student/s2i-go-app/Dockerfile
Application dockerfile generated in /home/student/s2i-go-app/Dockerfile
```

- 4.3. Compile una imagen de contenedor de prueba desde el Dockerfile generado.

```
[student@workstation custom-s2i]$ cd ~/s2i-go-app
[student@workstation s2i-go-app]$ sudo podman build -t s2i-go-app .
STEP 1: FROM s2i-do288-go
STEP 2: LABEL "io.k8s.display-name"="s2i-go-app"
...output omitted...
STEP 15: COMMIT s2i-go-app
```



### nota

Puede omitir de forma segura el siguiente error en la salida de compilación:  
**ERR0[0000] HOSTNAME is not supported for OCI image format,**  
**hostname 1d561c58fd2b will be ignored. Must use `docker`**  
**format** (Nombre de host no soportado para el formato de imagen OCI; el nombre  
de host 1d561c58fd2b se omitirá. Debe usar formato "docker"). Este es un error en  
la versión de la herramienta Podman y es un mensaje de error falso; la compilación  
del contenedor continúa sin problema a pesar del error. Puede encontrar más  
información en este Bugzilla [[https://bugzilla.redhat.com/show\\_bug.cgi?id=1661592](https://bugzilla.redhat.com/show_bug.cgi?id=1661592)]

- 4.4. Verifique que se haya creado la imagen del contenedor de aplicaciones:

```
[student@workstation s2i-go-app]$ sudo podman images
REPOSITORY TAG IMAGE ID ...
localhost/s2i-go-app latest 7d3d8f894f2f ...
...output omitted...
```

- 4.5. Pruebe la imagen del contenedor de aplicaciones localmente en la máquina virtual **workstation**. Ejecute el contenedor con el indicador **-u** para simular la ejecución como un usuario aleatorio en un clúster OpenShift. Puede copiar el comando o ejecutarlo directamente desde el script **~/DO288/labs/custom-s2i/local-test.sh**:

```
[student@workstation s2i-go-app]$ sudo podman run --name go-test -u 1234 \
> -p 8080:8080 -d s2i-go-app
18b903cfaae4...output omitted...
```

- 4.6. Verifique que el contenedor se inició correctamente:

```
[student@workstation s2i-go-app]$ sudo podman ps
CONTAINER ID IMAGE COMMAND
18b903cfaae4 localhost/s2i-go-app:latest /bin/sh -c /usr/lib...
```

- 4.7. Use el comando **curl** para probar la aplicación:

```
[student@workstation s2i-go-app]$ curl http://localhost:8080/user1
Hello user1!. Welcome!
```

- 4.8. Detenga el contenedor de la aplicación **go-test**:

```
[student@workstation s2i-go-app]$ sudo podman stop go-test
18b903cfaae4...output omitted...
[student@workstation s2i-go-app]$ cd ~
```

5. Envíe la imagen de compilador S2I **s2i-do288-go** a su cuenta personal de Quay.io.

- 5.1. 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
```

- 5.2. Inicie sesión en su cuenta de Quay.io con el comando **podman**. Ingrese su contraseña de Quay.io cuando se le solicite.

```
[student@workstation ~]$ sudo podman login \
> -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 5.3. Use el comando **skopeo copy** para publicar la imagen de compilador S2I en su cuenta de Quay.io.

```
[student@workstation ~]$ sudo skopeo copy \
> containers-storage:localhost/s2i-do288-go \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-go
...output omitted...
Writing manifest to image destination
Storing signatures
```

6. Cree un flujo de imágenes denominado s2i-do288-go para la imagen de compilador S2I s2i-do288-go. Cree el flujo de imágenes en un proyecto denominado *youruser-custom-s2i*.
  - 6.1. Inicie sesión en OpenShift y cree un proyecto nuevo. Coloque como prefijo del nombre del proyecto su nombre de usuario de desarrollador.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-custom-s2i
Now using project "youruser-custom-s2i" on server "https://
api.cluster.domain.example.com:6443".
```

- 6.2. Inicie sesión en su cuenta personal de Quay.io con Podman, esta vez sin el comando `sudo`, para poder exportar el archivo `auth.json` y usarlo en un secreto de extracción en el siguiente paso. Debe ingresar su contraseña nuevamente.

```
[student@workstation ~]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 6.3. Cree un secreto con el token de acceso de la API de registro de contenedores almacenado por Podman.  
También puede ejecutar o cortar y pegar el comando `oc create secret` siguiente del script `create-secret.sh` en el directorio `/home/student/D0288/labs/custom-s2i`.

```
[student@workstation ~]$ oc create secret generic quayio \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type=kubernetes.io/dockerconfigjson
secret/quayio created
```

- 6.4. Vincule el secreto nuevo con la cuenta de servicio `builder`.

```
[student@workstation ~]$ oc secrets link builder quayio
```

- 6.5. Cree un flujo de imágenes al importar la imagen de compilador S2I del registro privado del aula:

```
[student@workstation ~]$ oc import-image s2i-do288-go \
> --from quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-go \
> --confirm
imagestream.image.openshift.io/s2i-do288-go imported
...output omitted...
```

6.6. Verifique que se haya creado el flujo de imágenes:

```
[student@workstation ~]$ oc get is
NAME IMAGE REPOSITORY ...output omitted...
s2i-do288-go ...youruser-custom-s2i/s2i-do288-go ...output omitted...
```

7. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación master del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

8. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio y envíela a Github:

```
[student@workstation D0288-apps]$ git checkout -b custom-s2i
Switched to a new branch 'custom-s2i'
[student@workstation D0288-apps]$ git push -u origin custom-s2i
...output omitted...
 * [new branch] custom-s2i -> custom-s2i
Branch custom-s2i set up to track remote branch custom-s2i from origin.
[student@workstation D0288-apps]$ cd ~
```

9. Implemente y pruebe la aplicación go-hello desde su bifurcación personal de Github del repositorio D0288-apps al clúster OpenShift del aula. Asegúrese de hacer referencia a la bifurcación custom-s2i que creó en el paso anterior al implementar la aplicación. La aplicación devuelve un saludo al recurso solicitado por la solicitud HTTP. Por ejemplo, invocar la aplicación con la siguiente URL:

`http://greet-youruser-custom-s2i.apps.cluster.domain.example.com/user1`, devuelve la siguiente respuesta:

```
Hello user1!. Welcome!
```

- 9.1. Cree una nueva aplicación a partir del código fuente en Github y mediante el flujo de imágenes s2i-do288-go:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name greet \
> s2i-do288-go~https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#custom-s2i \
> --context-dir=go-hello
--> Found image a29d3e7 (About an hour old) in image stream "youruser-custom-s2i/
s2i-do288-go" under tag "latest" for "s2i-do288-go"

Go programming language S2I builder image for D0288
```

```
...output omitted...
--> Creating resources ...
 imagedstream.image.openshift.io "greet" created
 buildconfig.build.openshift.io "greet" created
 deploymentconfig.apps.openshift.io "greet" created
 service "greet" created
--> Success
...output omitted...
```

- 9.2. Vea los registros de compilación. Espere hasta que termine la compilación y la imagen del contenedor de aplicaciones se envíe al registro OpenShift:

```
[student@workstation ~]$ oc logs -f bc/greet
Cloning "https://github.com/youruser/D0288-apps" ...
...output omitted...
---> Installing application source...
---> Building application from source...
...output omitted...
Push successful
```

- 9.3. Espere hasta que la aplicación se haya implementado. Consulte el estado del pod de la aplicación. El pod de la aplicación debe estar en el estado Running (En ejecución):

```
[student@workstation ~]$ oc get pods
NAME READY STATUS RESTARTS AGE
greet-1-6zx8p 1/1 Running 0 35s
greet-1-build 0/1 Completed 0 2m30s
greet-1-deploy 0/1 Completed 0 43s
```

- 9.4. Exponga la aplicación a un acceso externo usando una ruta:

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

- 9.5. Obtenga la URL de enrutamiento con el comando `oc get route`:

```
[student@workstation ~]$ oc get route/greet -o jsonpath='{.spec.host}{"\n"}'
greet-youruser-custom-s2i.apps.cluster.domain.example.com
```

- 9.6. Pruebe la aplicación con la URL de ruta que obtuvo en el paso anterior:

```
[student@workstation ~]$ curl \
> http://greet-${RHT_OCP4_DEV_USER}-custom-s2i.${RHT_OCP4_WILDCARD_DOMAIN}/user1
Hello user1!. Welcome!
```

10. Personalice el script `run` para la imagen de compilador `s2i-do288-go` en la fuente de la aplicación. Cambie la manera en que se inicia la aplicación; para ello, agregue un argumento `--lang` es en el inicio. Esto cambia el idioma predeterminado que usa la aplicación.

Confirme y envíe los cambios a la bifurcación que usó como fuente de entrada para la compilación de la aplicación.

- 10.1. Los scripts S2I se pueden personalizar en el directorio `.s2i/bin` de la fuente de la aplicación ubicado en el directorio `D0288-apps/go-hello`. Cree el directorio:

```
[student@workstation ~]$ mkdir -p ~/D0288-apps/go-hello/.s2i/bin
```

- 10.2. Copie el script `run` en la imagen del compilador S2I de `~/D0288/labs/custom-s2i/s2i/bin/run`:

```
[student@workstation ~]$ cp ~/D0288/labs/custom-s2i/s2i/bin/run \
> ~/D0288-apps/go-hello/.s2i/bin/
```

- 10.3. Personalice el script `run` y agregue la opción `--lang es` en el inicio de la aplicación. También puede copiar el script completo del archivo `~/D0288/solutions/custom-s2i/s2i/bin/run.es`:

```
...output omitted...
echo "Starting app with lang option 'es'..."
exec /opt/app-root/app --lang es
```

- 10.4. Confirme los cambios en Git:

```
[student@workstation ~]$ cd ~/D0288-apps/go-hello
[student@workstation go-hello]$ git add .
[student@workstation go-hello]$ git commit -m "Customized run script"
...output omitted...
[student@workstation go-hello]$ git push
...output omitted...
[student@workstation go-hello]$ cd ~
```

11. Vuelva a compilar y pruebe la aplicación. La aplicación ahora debe responder a solicitudes en español. Por ejemplo, invocar la aplicación con la siguiente URL:

`http://greet-youruser-custom-s2i.apps.cluster.domain.example.com/user1`, devuelve la siguiente respuesta:

```
Hola user1!. Bienvenido!
```

- 11.1. Inicie una nueva compilación para la aplicación:

```
[student@workstation ~]$ oc start-build greet
build.build.openshift.io/greet-2 started
```

- 11.2. Siga el registro de compilación y verifique que se haya creado y se haya enviado una nueva imagen de contenedor al registro interno OpenShift:

```
[student@workstation ~]$ oc logs -f bc/greet
Cloning "https://github.com/youruser/D0288-apps" ...
Commit: 0023a1b02342b633aad49e58a2eeba11dff33c3d (customized run script)
...output omitted...
Push successful
```

- 11.3. Espere a que se implemente el pod de la aplicación. El pod debe estar en el estado **Running** (En ejecución). Verifique el estado del pod de la aplicación:

```
[student@workstation ~]$ oc get pods
NAME READY STATUS RESTARTS AGE
...output omitted...
greet-2-build 0/1 Completed 0 2m14s
greet-2-deploy 1/1 Running 0 44s
greet-2-rpwb4 1/1 Running 0 34s
```

- 11.4. Pruebe la aplicación con la URL de ruta que obtuvo en el *Paso 9.5*:

```
[student@workstation ~]$ curl \
> http://greet-${RHT_OCP4_DEV_USER}-custom-s2i.${RHT_OCP4_WILDCARD_DOMAIN}/user1
Hola user1!. Bienvenido!
```

- 12.** Califique su trabajo.

Ejecute el siguiente comando en la máquina virtual **workstation** para verificar que se hayan realizado todas las tareas:

```
[student@workstation ~]$ lab custom-s2i grade
```

- 13.** Realice la limpieza. Realice los siguientes pasos:

- 13.1. Elimine el proyecto **youruser-custom-s2i** de OpenShift.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-custom-s2i
```

- 13.2. Elimine todos los contenedores de prueba creados antes para probar la aplicación localmente.

```
[student@workstation ~]$ sudo podman rm go-test
```

- 13.3. Elimine todas las imágenes de contenedor compiladas durante este trabajo de laboratorio en la máquina virtual **workstation**.

```
[student@workstation ~]$ sudo podman rmi -f \
> localhost/s2i-go-app \
> localhost/s2i-do288-go \
> registry.access.redhat.com/ubi8/ubi:8.0
...output omitted...
```

- 13.4. Elimine la imagen **s2i-do288-go** del registro externo:

```
[student@workstation ~]$ sudo skopeo delete \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/s2i-do288-go:latest
```

- 13.5. Inicie sesión en **Quay.io** con su cuenta personal gratuita.

Desplácese hasta **http://quay.io** y haga clic en **Sign In** (Iniciar sesión) para proporcionar sus credenciales de usuario. Haga clic en **Sign in to Quay Container**

- Registry (Iniciar sesión en el registro de contenedor de Quay) para iniciar sesión en Quay.io.
- 13.6. En el menú principal de Quay.io, haga clic en **Repositories** (Repositorios) y busque **s2i-do288-go**. El icono de candado indica que es un repositorio privado que requiere autenticación para extracciones e inserciones. Haga clic en **s2i-do288-go** para mostrar la página **Repository Activity** (Actividad del repositorio).
  - 13.7. En la página **Repository Activity** (Actividad del repositorio) del repositorio **s2i-do288-go**, desplácese hacia abajo y haga clic en el ícono de engranaje para mostrar la pestaña **Settings** (Configuración). Desplácese hacia abajo y haga clic en **Delete Repository** (Eliminar repositorio).
  - 13.8. En el cuadro de diálogo **Delete** (Eliminar), haga clic en **Delete** (Eliminar) para confirmar que desea eliminar el repositorio **s2i-do288-go**. Después de unos momentos, volverá a la página **Repositories** (Repositorios). Ahora puede cerrar sesión en Quay.io.

## Finalizar

En la máquina virtual **workstation**, ejecute el comando **lab custom-s2i finish** para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab custom-s2i finish
```

Esto concluye el trabajo de laboratorio.

# Resumen

---

En este capítulo, aprendió lo siguiente:

- Una imagen de compilador S2I es una imagen de contenedor especializada que usan los desarrolladores para producir imágenes de contenedores de aplicaciones. Las imágenes de compilador incluyen librerías del sistema operativo base, tiempos de ejecución de lenguaje, marcos (frameworks) y dependencias de aplicaciones, y utilidades y herramientas de fuente a imagen.
- Una imagen de compilador S2I proporciona scripts S2I de manera predeterminada. Estos scripts S2I se pueden sobreescibir en el código fuente de la aplicación al agregar scripts S2I al directorio `.s2i/bin`.
- La herramienta de línea de comandos `s2i` se usa para compilar y probar las imágenes de compilador S2I fuera de OpenShift.
- En entornos RHEL 8 u OpenShift 4, donde Docker no está disponible de forma predeterminada, use la opción `--as-dockerfile` para el comando `s2i build`. Esto hace que el comando genere un Dockerfile y directorios auxiliares que puede compilar con Podman para probar la imagen de compilador S2I.



## capítulo 6

# Creación de aplicaciones desde plantillas de OpenShift

### Meta

Describir los elementos de una plantilla y crear una plantilla de aplicación con varios contenedores.

### Objetivos

- Describir los elementos de una plantilla de OpenShift.
- Compilar una aplicación con varios contenedores desde una plantilla personalizada.

### Secciones

- Descripción de los elementos de una plantilla de OpenShift (y cuestionario)
- Creación de una plantilla con varios contenedores (y ejercicio guiado)

### Trabajo de laboratorio

Creación de aplicaciones desde plantillas de OpenShift

# Descripción de los elementos de una plantilla de OpenShift

---

## Objetivos

Después de completar esta sección, deberá ser capaz de describir los elementos de una plantilla de OpenShift.

## Descripción de una plantilla

Una *plantilla* OpenShift es un archivo YAML o JSON que consta de un conjunto de recursos de OpenShift. Las plantillas definen los *parámetros* que se usan para personalizar la configuración de recursos. OpenShift procesa plantillas reemplazando referencias de parámetros con valores y creando un conjunto personalizado de recursos.

Las plantillas de OpenShift ofrecen una funcionalidad similar a los gráficos de Kubernetes Helm. Puede usar gráficos Helm o plantillas de OpenShift para administrar sus aplicaciones de varios contenedores de forma similar. Sin embargo, los gráficos Helm no están en el ámbito de este curso.

Una plantilla es útil cuando desea implementar un conjunto de recursos como una sola unidad, en lugar de implementarlos individualmente. Estos son algunos casos de uso de ejemplo para saber cuándo usar una plantilla:

- Un proveedor de software independiente (ISV) proporciona una plantilla para implementar su producto en OpenShift. La plantilla contiene detalles de configuración de los contenedores que componen el producto, junto con una configuración de implementación, como el número de réplicas, servicios y rutas, la configuración de almacenamiento persistente, las comprobaciones de estado y los límites de recursos para los recursos de cómputo como CPU, memoria y E/S.
- La aplicación multinivel consta de una serie de componentes independientes, como un servidor web, un servidor de aplicaciones y una base de datos, y le gustaría implementar estos componentes juntos como una sola unidad en OpenShift para simplificar el proceso de implementación de la aplicación en un entorno de ensayo. Esto permitirá a sus equipos de pruebas de control de calidad y de aceptación aprovisionen rápidamente aplicaciones para pruebas.

## Sintaxis de la plantilla

La sintaxis de una plantilla OpenShift sigue la sintaxis general de un recurso de OpenShift, pero con el atributo `objects` (objetos) en lugar del atributo `spec` (especificación). Una plantilla generalmente incluye atributos de `parameters` (parámetros) y `labels` (etiquetas), así como anotaciones en sus metadatos.

En la siguiente lista, se muestra una definición de plantilla de prueba en formato YAML y se ilustran los elementos de la sintaxis principal. Como con cualquier recurso de OpenShift, también puede definir plantillas en la sintaxis JSON:

```
apiVersion: template.openshift.io/v1
kind: Template ①
metadata:
 name: mytemplate
```

```

annotations:
 description: "Description" ❷
objects: ❸
- apiVersion: v1
 kind: Pod
 metadata:
 name: myapp
 spec:
 containers:
 - env:
 - name: MYAPP_CONFIGURATION
 value: ${MYPARAMETER} ❹
 image: myorganization/myapplication
 name: myapp
 ports:
 - containerPort: 80
 protocol: TCP
 parameters: ❺
 - description: Myapp configuration data
 name: MYPARAMETER
 required: true
 labels: ❻
 mylabel: myapp

```

- ❶ Tipo de recurso de plantilla
- ❷ Anotaciones opcionales para su uso con las herramientas de OpenShift
- ❸ Lista de recursos
- ❹ Referencia a un parámetro de plantilla
- ❺ Lista de parámetros
- ❻ Lista de etiquetas

La lista de recursos de una plantilla normalmente incluye otros recursos como configuraciones de compilación, configuraciones de implementación, reclamaciones de volumen persistente (PVC), servicios y rutas.

Todo atributo en la lista de recursos puede hacer referencia al valor de cualquier parámetro.

Puede definir etiquetas personalizadas para una plantilla. OpenShift agrega estas etiquetas a todos los recursos creados por la plantilla.

Cuando una plantilla define varios recursos, es esencial considerar el orden en el que se definen estos recursos para alojar dependencias entre recursos. OpenShift no informa un error si un recurso hace referencia a un recurso dependiente que no existe.

Un proceso desencadenado por un recurso puede fallar si se lo inicia antes del recurso dependiente. Un ejemplo es una configuración de compilación que hace referencia a una secuencia de imágenes como la imagen de salida y la plantilla define esa secuencia de imágenes después de la configuración de compilación. En este escenario, es posible que la configuración de compilación inicie una compilación antes de que exista la secuencia de imágenes.

## Definición de parámetros de plantilla

La plantilla del ejemplo anterior incluyó la definición de un parámetro requerido. Tanto los parámetros opcionales como los requeridos pueden ofrecer valores predeterminados. Por ejemplo:

```
parameters:
- description: Myapp configuration data
 name: MYPARAMETER
 value: /etc/myapp/config.ini
```

OpenShift puede generar valores predeterminados aleatorios para parámetros. Esto es útil para secretos y contraseñas:

```
parameters:
- description: ACME cloud provider API key
 name: APIKEY
 generate: expression
 from:"[a-zA-Z0-9]{12}"
```

La sintaxis para valores generados es un subconjunto de sintaxis de expresión regular Perl. Vea las referencias al final de esta sección para consultar la sintaxis completa de expresión de parámetros de la plantilla.

## Agregar una plantilla a OpenShift

Para agregar una plantilla a OpenShift, use el comando `oc create` o la consola web (con la página de importación de YAML) para crear el recurso de plantilla desde el archivo de definición de plantilla, de la misma forma que con cualquier otro recurso.

Es una práctica común crear proyectos que contienen solo plantillas, porque OpenShift permite que los usuarios compartan fácilmente las plantillas entre varios usuarios y proyectos. Estos proyectos suelen incluir otros posibles recursos compartidos, como flujos de imágenes.

Una instalación predeterminada de Red Hat OpenShift Container Platform ofrece varias plantillas en el proyecto `openshift`. Todos los usuarios del clúster OpenShift tienen acceso de lectura al proyecto `openshift`, pero solo los administradores de clústeres tienen permiso para crear o eliminar plantillas en este proyecto.

## Exploración de plantillas con la CLI de OpenShift

OpenShift proporciona una serie de plantillas en el espacio de nombres `openshift` de forma predeterminada. Puede crear aplicaciones a partir de estas plantillas, así como personalizar las plantillas proporcionadas para que se adapten a las necesidades de la aplicación.

Para ver la lista de plantillas proporcionadas, use el comando `oc get templates`:

```
[user@host ~]$ oc get templates -n openshift
```

Para ver la lista de parámetros proporcionada por la plantilla, así como una breve descripción, use el comando `oc process`. Por ejemplo, para ver los parámetros de una plantilla Node.js y MongoDB proporcionada por OpenShift:

```
[user@host ~]$ oc process --parameters -n openshift nodejs-mongodb-example
```

Para exportar la plantilla como un archivo YAML para usarlo como base para su propia plantilla personalizada, use el comando `oc get`. Por ejemplo, para exportar la plantilla `nodejs-mongodb-example` proporcionada por OpenShift en formato YAML:

```
[user@host ~]$ oc get template nodejs-mongodb-example -o yaml -n openshift
```

También puede usar el comando `oc describe template` para obtener una descripción más detallada de una plantilla. Por ejemplo, para describir la plantilla `nodejs-mongodb-example` proporcionada por OpenShift:

```
[user@host ~]$ oc describe template nodejs-mongodb-example -n openshift
 Name: nodejs-mongodb-example 1
 Namespace: openshift
 Created: 6 days ago
 Labels: samples.operator.openshift.io/managed=true
...output omitted...
 Annotations: iconClass=icon-nodejs
 openshift.io/display-name=Node.js + MongoDB (Ephemeral)
 openshift.io/documentation-url=https://github.com/sclorg/nodejs-ex
...output omitted...
 Parameters: 2
 Name: NAME
 Display Name: Name
 Description: The name assigned to all of the frontend objects defined in
this template.
 Required: true
 Value: nodejs-mongodb-example

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

 Name: NODEJS_VERSION
 Display Name: Version of NodeJS Image
 Description: Version of NodeJS image to be used (6, 8, or latest).
 Required: true
 Value: 8
...output omitted...
 Object Labels: app=nodejs-mongodb-example,template=nodejs-mongodb-example 3

 Message: The following service(s) have been created in your project: ${NAME},
${DATABASE_SERVICE_NAME}.

 For more information about using this template, including OpenShift
considerations, see https://github.com/sclorg/nodejs-ex/blob/master/README.md.

 Objects: 4
 Secret ${NAME}
 Service ${NAME}
 Route ${NAME}
 ImageStream ${NAME}
 BuildConfig ${NAME}
 DeploymentConfig ${NAME}
 Service ${DATABASE_SERVICE_NAME}
 DeploymentConfig ${DATABASE_SERVICE_NAME}
```

- ① El nombre de la plantilla.
- ② Una lista de parámetros que ofrece la plantilla, incluidos si son obligatorios y los valores predeterminados.
- ③ Una lista de etiquetas que OpenShift aplica a todos los recursos que crea a partir de la plantilla.
- ④ Un resumen de todos los recursos que OpenShift crea a partir de la plantilla.



### Referencias

Para encontrar más información, consulte el capítulo *Uso de plantillas* en la documentación para Red Hat OpenShift Container Platform 4.5 en  
[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.5/html/images/using-templates](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/images/using-templates)

## ► Cuestionario

# Descripción de los elementos de una plantilla de OpenShift

Elija las respuestas correctas para las siguientes preguntas:

► 1. ¿Cuál es el propósito de una plantilla de OpenShift?

- a. Describir la configuración de recursos para un único contenedor.
- b. Encapsular un recurso de OpenShift para reutilización.
- c. Ofrecer opciones de configuración personalizadas para un clúster OpenShift.
- d. Personalizar la apariencia de la consola web.

► 2. ¿Cuáles dos de los siguientes enfoques definen un parámetro de plantilla como opcional? (Elija dos opciones).

- a. Establezca el atributo required como false.
- b. Establezca el atributo required en not.
- c. Omita el atributo required.
- d. Establezca el atributo required en expression.
- e. Establezca el atributo from como una expresión regular.

► 3. ¿Cuál de los siguientes comandos devuelve el número de parámetros y objetos proporcionados por una plantilla?

- a. oc export.
- b. oc describe.
- c. oc create.
- d. oc get.
- e. Ninguna de las anteriores.

► 4. ¿Cuáles tres de las siguientes declaraciones describen ubicaciones válidas para insertar referencias de parámetros de plantillas? (Elija tres opciones).

- a. Como el nombre de una variable de entorno dentro de un recurso de pod.
- b. Como la clave para una etiqueta en todos los recursos creados por la plantilla.
- c. Como el valor para el atributo host en un recurso de ruta.
- d. Como el valor para una anotación en la plantilla.
- e. Como el valor de una variable de entorno dentro de un recurso de pod.

## ► Solución

# Descripción de los elementos de una plantilla de OpenShift

Elija las respuestas correctas para las siguientes preguntas:

► 1. ¿Cuál es el propósito de una plantilla de OpenShift?

- a. Describir la configuración de recursos para un único contenedor.
- b. Encapsular un recurso de OpenShift para reutilización.
- c. Ofrecer opciones de configuración personalizadas para un clúster OpenShift.
- d. Personalizar la apariencia de la consola web.

► 2. ¿Cuáles dos de los siguientes enfoques definen un parámetro de plantilla como opcional? (Elija dos opciones).

- a. Establezca el atributo **required** como **false**.
- b. Establezca el atributo **required** en **not**.
- c. Omita el atributo **required**.
- d. Establezca el atributo **required** en **expression**.
- e. Establezca el atributo **from** como una expresión regular.

► 3. ¿Cuál de los siguientes comandos devuelve el número de parámetros y objetos proporcionados por una plantilla?

- a. **oc export**.
- b. **oc describe**.
- c. **oc create**.
- d. **oc get**.
- e. Ninguna de las anteriores.

► 4. ¿Cuáles tres de las siguientes declaraciones describen ubicaciones válidas para insertar referencias de parámetros de plantillas? (Elija tres opciones).

- a. Como el nombre de una variable de entorno dentro de un recurso de pod.
- b. Como la clave para una etiqueta en todos los recursos creados por la plantilla.
- c. Como el valor para el atributo **host** en un recurso de ruta.
- d. Como el valor para una anotación en la plantilla.
- e. Como el valor de una variable de entorno dentro de un recurso de pod.

# Creación de una plantilla con varios contenedores

## Objetivos

Tras finalizar esta sección, deberá ser capaz de compilar una aplicación con varios contenedores a partir de una plantilla personalizada.

## Creación de una plantilla a partir de recursos de aplicación

Un escenario típico para el uso de plantillas de OpenShift es implementar una aplicación que requiere varios conjuntos de pods, cada uno con una configuración de implementación, servicios y otros recursos auxiliares, como secretos y reclamaciones de volúmenes persistentes.

Considere una aplicación que se conecta a una base de datos para almacenar y administrar datos. Una plantilla para esta aplicación incluiría:

- Una configuración de implementación y un servicio para la aplicación, además de otra configuración de implementación y un servicio para la base de datos.
- Dos flujos de imágenes para indicar las imágenes de contenedor: uno para la aplicación y otro para la base de datos.
- Un secreto para las credenciales de acceso de la base de datos.
- Una reclamación de volumen persistente para almacenar los datos de la base de datos.
- Una ruta para acceso externo a la aplicación.

Puede crear la plantilla requerida de varias maneras. Los enfoques más comunes se describen en la siguiente sección.

## Creación manual de archivos de recursos

Si está cómodo con la sintaxis de los archivos YAML o JSON y está familiarizado con los atributos para cada uno de los tipos de recursos de OpenShift, puede crear plantillas a mano.

El proceso general para este enfoque se describe a continuación.

1. Cree una plantilla esqueleto básica con atributos básicos como un nombre, etiquetas y otra información de metadatos y déclarélo como un recurso **Template** (Plantilla) de OpenShift.
2. Observe las plantillas existentes que vienen instaladas con OpenShift de forma predeterminada y personalícelas según sus necesidades cortando y pegando los recursos relevantes para su aplicación.
3. Para personalizar aún más la plantilla, use los comandos `oc explain` y `oc api-resources` para ver los atributos de cada tipo de recurso que desee personalizar y, a continuación, agréguelo a la plantilla.

## Concatenación de archivos de recursos

El comando `oc new-app` puede crear un archivo de definición de recursos, en lugar de crear recursos en el proyecto actual, al usar la opción `-o`. Puede concatenar varios archivos de

definición de recursos dentro de la lista de recursos de un archivo de definición de la plantilla esqueleto.

El proceso general para este enfoque se describe a continuación.

1. Cree una plantilla esqueleto básica con atributos básicos como un nombre, etiquetas y otra información de metadatos y déclarélo como un recurso **Template** (Plantilla) de OpenShift.
2. Use el comando `oc new-app` con la opción `-o` para exportar la configuración de recursos a un archivo.
3. Limpie los atributos de tiempo de ejecución del archivo de recursos generado.
4. Corte y pegue la lista de recursos en un archivo de plantilla esqueleto.

## Exportación de recursos existentes

El comando `oc get` puede crear un archivo de definición de recursos al usar las opciones `-o` y `--export` juntas. La opción `-o` toma `yaml` o `json` como parámetro y exporta la definición de recurso en formato YAML o JSON, respectivamente. Después de exportar un conjunto de recursos a un archivo de plantilla, puede agregar anotaciones y parámetros según desee.

El proceso general para este enfoque se describe a continuación.

1. Cree una plantilla esqueleto básica con atributos básicos como un nombre, etiquetas y otra información de metadatos y déclarélo como un recurso **Template** (Plantilla) de OpenShift.
2. Use el comando `oc get` con las opciones `-o --export` para exportar la configuración de recursos a un archivo.
3. Limpie los atributos de tiempo de ejecución del archivo de recursos generado.
4. Corte y pegue la lista de recursos en un archivo de plantilla esqueleto.

Si usa el comando `oc get` para exportar recursos, seleccione solo los tipos de recurso necesarios. Incluya únicamente recursos que necesitaría para implementar una aplicación. No incluya recursos derivados, como compilaciones, implementaciones, controladores de replicación o pods.

El orden en el que enumera los recursos en el comando `oc get` es importante. Primero debe exportar los recursos dependientes y, luego, los recursos que dependen de ellos. Por ejemplo, necesita exportar flujos de imágenes antes de las opciones de configuración de compilación y opciones de configuración de implementación que hacen referencia a esos flujos de imágenes.

En el ejemplo siguiente, se crea un archivo YAML que contiene diferentes tipos de recursos en el proyecto actual:

```
[user@host ~]$ oc get -o yaml --export is,bc,dc,svc,route > mytemplate.yaml
```

Según sus necesidades, agregue más tipos de recursos al comando anterior. Por ejemplo, agregue `secret` antes de `bc` y `dc`. Es seguro agregar `pvc` al final de la lista de tipos de recurso, ya que una implementación espera una reclamación de volumen persistente para vincular.

## Limpieza y edición de plantillas

Los comandos `oc get` y `oc new-app` no generan definiciones de recursos que están listos para usarse en una plantilla. Estas definiciones de recursos contienen información de tiempo de ejecución que una plantilla no necesita y parte de esta podría impedir que la plantilla funcione en absoluto. Atributos como `status`, `creationTimeStamp`, `image` y `uid` son algunos ejemplos de

información de tiempo de ejecución, además de la mayoría de las anotaciones que comienzan con el prefijo `openshift.io/generated-by`.

Algunos tipos de recursos, como los secretos, requieren un manejo especial. No es posible inicializar valores clave dentro del atributo `data` con parámetros de plantilla. El atributo `data` de un recurso secreto se debe reemplazar por el atributo `stringData` y todos los valores clave deben decodificarse.

Otro ejemplo son los recursos de flujo de imágenes. OpenShift almacena en caché las imágenes de contenedor de los registros externos mediante el registro interno, y el flujo de imágenes exportado apunta al registro interno. Si usa este recurso de flujo de imágenes en una plantilla, fallará, ya que la imagen de contenedor no existe en el proyecto actual. Debe cambiar el flujo de imágenes para que indique el registro externo.

## Descripción de tipos de recursos de OpenShift

Para crear una plantilla de cero o limpiar un archivo de recursos generado por el comando `oc get`, debe saber qué atributos contienen información de tiempo de ejecución. Esto se explica en la documentación del producto Red Hat OpenShift Container Platform. Consulte las referencias al final de esta sección. Otro método consiste en usar el comando `oc explain`.

Con un tipo de recurso como argumento, el comando `oc explain` enumera los atributos de nivel superior para ese tipo de recurso:

```
[user@host ~]$ oc explain routes
DESCRIPTION:
A route allows developers to expose services through an HTTP(S) aware load
balancing and proxy layer via a public DNS entry.
...output omitted...
FIELDS:
 apiVersion <string>
 ...output omitted...

 kind <string>
 ...output omitted...

 metadata <Object>
 Standard object metadata.

 spec <Object> -required-
 spec is the desired state of the route
 ...output omitted...
```

La mayoría de los tipos de recursos tienen muchos niveles de atributos. Use la notación de punto para solicitar atributos de niveles inferiores:

```
[user@host ~]$ oc explain routes.spec
RESOURCE: spec <Object>

DESCRIPTION:
 spec is the desired state of the route
 ...output omitted...
FIELDS:
 to <Object> -required-
```

```

to is an object the route should use as the primary backend.
...output omitted...

wildcardPolicy <string>
 Wildcard policy if any for the route. Currently only 'Subdomain' or 'None'
 is allowed.
...output omitted...

```

Use el comando `oc api-resources` para obtener una lista completa de los recursos soportados en la instancia de OpenShift.

## Creación de una aplicación a partir de una plantilla

Puede implementar una aplicación directamente desde un archivo de definición de recursos de plantilla. El comando `oc new-app` y el comando `oc process` usan el archivo de plantilla como entrada y lo procesan para aplicar parámetros y crear recursos.

También puede pasar un archivo de plantilla al comando `oc create` para crear un recurso de plantilla en OpenShift. Si se pretende reutilizar la plantilla por varios desarrolladores, se recomienda crear el recurso de la plantilla en un proyecto compartido. Si se pretende usar la plantilla mediante una única implementación, se recomienda mantenerla en un archivo.

Mientras que el comando `oc new-app` crea recursos desde la plantilla, el comando `oc process` crea una lista de la plantilla. Debe guardar la lista de recursos generada por el comando `oc process` en un archivo o enviarla como entrada al comando `oc create` para crear los recursos de la lista.

Tanto para el comando `oc new-app` como para el comando `oc process`, cada valor de parámetro se debe ofrecer usando una opción `-p` diferente. Otra opción que aceptan ambos comandos es la opción `-l`, la cual agrega una etiqueta a todos los recursos creados a partir de la plantilla.

El siguiente es un ejemplo de comando `oc new-app` que implementa una aplicación de un archivo de plantilla:

```
[user@host ~]$ oc new-app --file mytemplate.yaml -p PARAM1=value1 \
> -p PARAM2=value2
```

El siguiente es un ejemplo de comando `oc process` que implementa una aplicación de un archivo de plantilla:

```
[user@host ~]$ oc process -f mytemplate.yaml -p PARAM1=value1 \
> -p PARAM2=value2 > myresourcelist.yaml
```

A continuación, el archivo generado por el ejemplo anterior se entrega al comando `oc create`:

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

Puede combinar los dos ejemplos anteriores con una tubería:

```
[user@host ~]$ oc process -f mytemplate.yaml -p PARAM1=value1 \
> -p PARAM2=value2 | oc create -f -
```

Red Hat recomienda usar el comando `oc new-app` en lugar del comando `oc process`. Puede usar el comando `oc process` con la opción `-f` para enumerar solo los parámetros definidos por la plantilla especificada:

```
[user@host ~]$ oc process -f mytemplate.yaml --parameters
```



### nota

La consola web de OpenShift 4.5 no contiene ninguna sección para mostrar o ver las plantillas. Sin embargo, puede mostrar las plantillas disponibles en las páginas **Administrator** → **Home** → **Search** (Administrador > Inicio > Buscar) y **Developer** → **Search** (Desarrollador > Buscar).

De manera predeterminada, no puede crear aplicaciones a partir de plantillas personalizadas en la consola web de OpenShift 4.5. Los administradores de clústeres pueden instalar plantillas adicionales en el catálogo del desarrollador. Las plantillas en el catálogo del desarrollador se pueden usar para crear nuevas aplicaciones.



### Referencias

Para encontrar más información sobre la creación y el uso de plantillas, consulte el capítulo *Uso de plantillas* en la documentación de Red Hat OpenShift Container Platform en

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.5/html/images/using-templates](https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/images/using-templates)

## ► Ejercicio Guiado

# Creación de una plantilla con varios contenedores

En este ejercicio, creará una plantilla de OpenShift que implementa una aplicación de varios contenedores. La plantilla incluye una API HTTP, una base de datos y un almacenamiento persistente.

## Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear una plantilla de una aplicación en ejecución mediante el comando `oc get`.
- Limpiar la plantilla para eliminar información de tiempo de ejecución.
- Agregar parámetros a la plantilla.
- Crear una nueva aplicación a partir de la plantilla con la herramienta de línea de comandos de OpenShift (`oc`).

## Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador S2I PHP 7.2 y la imagen de la base de datos MySQL 5.7 requeridas por la plantilla.
- La aplicación de muestra (quotes) en el repositorio Git.

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos, aprovisione el almacenamiento persistente, implemente la aplicación con varios contenedores en el proyecto `youruser-quotes-dev` y descargue los archivos requeridos para completar este ejercicio:

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

Par revisar cómo se implementó la aplicación, consulte los scripts `new-app-db.sh`, `add-vol.sh`, `new-app-php.sh` y `add-route.sh` en la carpeta `~/DO288/labs/create-template`.

### ► 1. Inspeccione la aplicación quotes implementada en el proyecto `youruser-quotes-dev`.

- 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 OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

- 1.3. Establezca *youruser-quotes-dev* como su proyecto activo actual:

```
[student@workstation ~]$ oc project ${RHT_OCP4_DEV_USER}-quotes-dev
```

- 1.4. La base de datos y la aplicación ya están implementadas. Espere a que se complete la compilación y los pods de la aplicación y la base de datos estén listos y en ejecución:

```
[student@workstation ~]$ oc status
In project youruser-quotes-dev on server
https://api.cluster.domain.example.com:6443

http://quotesapi-youruser-quotes-dev.apps.cluster.domain.example.com to pod port
8080-tcp (svc/quotesapi)
dc/quotesapi deploys istag/quotesapi:latest <-
bc/quotesapi source builds https://github.com/youruser/D0288-apps on openshift/
php:7.2
deployment #1 deployed 28 seconds ago - 1 pod

svc/quotesdb - 172.30.239.22:3306
dc/quotesdb deploys openshift/mysql:5.7
deployment #2 deployed about a minute ago - 1 pod
deployment #1 failed about a minute ago: newer deployment was found running
...output omitted...
```

En la salida anterior se muestra que la base de datos se implementa de una imagen de contenedor, y la aplicación se implementa de un código fuente.

- 1.5. Verifique que el proyecto incluya una reclamación de volumen persistente:

```
[student@workstation ~]$ oc get pvc
NAME STATUS VOLUME CAPACITY ACCESS MODES ...
quotesdb-claim Bound pvc-df1c... 1Gi RWO ...
```

- 1.6. Verifique que el proyecto incluya una ruta:

```
[student@workstation ~]$ oc get route/quotesapi -o jsonpath='{.spec.host}{ "\n" }'
quotesapi-youruser-quotes-dev.apps.cluster.domain.example.com
```

No intente probar la aplicación. Fallará porque no se ha inicializado la base de datos. El único uso del proyecto *youruser-quotes-dev* es exportar sus recursos a una plantilla.

- 2. Cree el archivo de definición de la plantilla. Comience con un archivo de definición de plantilla básico y, a continuación, exporte los recursos necesarios uno por uno, límpie la información de tiempo de ejecución y, a continuación, copie la configuración de recursos limpia en la plantilla.

Cree un nuevo archivo llamado `quotes-template-clean.yaml` en el directorio `/home/student`. Agregue el siguiente fragmento de código YAML para declarar este archivo de definición de recursos como una plantilla de OpenShift:

```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
 name: quotes
 annotations:
 openshift.io/display-name: Quotes Application
 description: The Quotes application provides an HTTP API that returns a
 random, funny quote.
 iconClass: icon-php
 tags: php,mysql
objects:
```

También puede copiar el fragmento de código YAML del archivo `~/D0288/solutions/create-template/new-template.yaml`.



### Advertencia

Asegúrese de que el valor del atributo `description` esté en una sola línea continua sin saltos de línea. Obtendrá errores por archivos YAML con formato incorrecto cuando la plantilla se analice más adelante en el ejercicio.

- ▶ 3. Exporte los recursos del proyecto uno por uno, empezando por los recursos de flujo de imágenes.

- 3.1. Exporte el flujo de imágenes:

```
[student@workstation ~]$ oc get -o yaml --export is > /tmp/is.yaml
```

- 3.2. Haga una copia del archivo exportado y limpie los atributos de tiempo de ejecución de este:

```
[student@workstation ~]$ cp /tmp/is.yaml /tmp/is-clean.yaml
```

- 3.3. Limpie los atributos de tiempo de ejecución del archivo `/tmp/is-clean.yaml`.

Una copia del archivo limpiado está disponible en `~/D0288/labs/create-template/is-clean.yaml`. Compare el archivo limpio con esta versión y realice las ediciones necesarias para que sean iguales.

Elimine las primeras dos líneas del archivo:

```
apiVersion: v1
items:
```

- 3.4. Hay dos recursos de flujo de imágenes en el archivo exportado. El primero es para la aplicación `quotesapi` y el segundo para la base de datos `quotesdb`. Quite todo el recurso de flujo de imágenes para la base de datos `quotesdb`. La configuración de implementación de la base de datos creará el flujo de imágenes automáticamente.

Comenzando con la segunda aparición del recurso de flujo de imágenes, es decir:

```

- apiVersion: image.openshift.io/v1
kind: ImageStream
metadata:
 annotations:
 openshift.io/image.dockerRepositoryCheck: "2019-06-21T02:54:22Z"
 creationTimestamp: "2019-06-21T02:54:22Z"
 generation: 2
 name: quotesdb

...output omitted...

```

Elimine todas las líneas desde el fragmento de código mencionado anteriormente hasta el final del archivo.

- 3.5. Los pasos siguientes implican la limpieza del recurso de flujo de imágenes solo para la aplicación quotesapi.  
Elimine los atributos `openshift.io/generated-by`, `creationTimestamp`, `generation`, `namespace`, `resourceVersion`, `selfLink` y `uid`.  
Elimine los atributos `managedFields` y `status`, incluidos sus atributos secundarios.

#### ▶ 4. Exporte y limpie la configuración de compilación.

- 4.1. Exporte la configuración de compilación:

```
[student@workstation ~]$ oc get -o yaml --export bc > /tmp/bc.yaml
```

- 4.2. Haga una copia del archivo exportado y limpie los atributos de tiempo de ejecución de este:

```
[student@workstation ~]$ cp /tmp/bc.yaml /tmp/bc-clean.yaml
```

Limpie los atributos de tiempo de ejecución del archivo `/tmp/bc-clean.yaml`.  
Una copia del archivo limpiado está disponible en `~/D0288/labs/create-template/bc-clean.yaml`. Compare el archivo limpio con esta versión y realice las ediciones necesarias para que sean iguales.  
Ignore las diferencias en los atributos `secret` y `uri`. Estos valores serán diferentes para usted.

- 4.3. Elimine las primeras dos líneas del archivo:

```

apiVersion: v1
items:

```

- 4.4. Elimine los atributos `openshift.io/generated-by`, `creationTimestamp`, `generation`, `namespace`, `resourceVersion`, `selfLink` y `uid`.
- 4.5. Elimine el atributo `managedFields` y todos sus atributos secundarios.
- 4.6. Elimine el atributo `namespace` que hace referencia al proyecto `youruser-quotes-dev`. No elimine la referencia `namespace`: `openshift` debajo del atributo `sourceStrategy` más abajo en el archivo.

- 4.7. Elimine el atributo `lastTriggeredImageID` en `imageChange`.
- 4.8. Elimine el atributo `status` y todos sus atributos secundarios al final del archivo.
- 4.9. Elimine el atributo `kind: List` al final del archivo y todos sus atributos secundarios.

► 5. Exporte y limpie la configuración de implementación para la aplicación `quotesapi` y la base de datos `quotesdb`.

- 5.1. Exporte la configuración de implementación:

```
[student@workstation ~]$ oc get -o yaml --export dc > /tmp/dc.yaml
```

- 5.2. Haga una copia del archivo exportado y limpie los atributos de tiempo de ejecución de este:

```
[student@workstation ~]$ cp /tmp/dc.yaml /tmp/dc-clean.yaml
```

- 5.3. Limpie los atributos de tiempo de ejecución del archivo `/tmp/dc-clean.yaml`. Una copia del archivo limpiado está disponible en `~/D0288/labs/create-template/dc-clean.yaml`. Compare el archivo limpio con esta versión y realice las ediciones necesarias para que sean iguales.

- 5.4. Elimine las primeras dos líneas del archivo:

```
apiVersion: v1
items:
```

- 5.5. Elimine todas las referencias a los atributos `openshift.io/generated-by`, `creationTimestamp`, `generation`, `resourceVersion`, `selfLink` y `uid` del archivo.
- 5.6. Elimine el atributo `namespace` que hace referencia al proyecto `youruser-quotes-dev`. No elimine la referencia `namespace: openshift` debajo del atributo `imageChangeParams` más abajo en el archivo.
- 5.7. Elimine el atributo `managedFields` y todos sus atributos secundarios.
- 5.8. Elimine todas las referencias a los atributos `image` y `lastTriggeredImage` del archivo.
- 5.9. Elimine el atributo `status` y todos sus atributos secundarios para ambos recursos de configuración de implementación.
- 5.10. Elimine el atributo `kind: List` al final del archivo y todos sus atributos secundarios.

► 6. Exporte y limpie la configuración de servicio para la aplicación `quotesapi` y la base de datos `quotesdb`.

- 6.1. Exporte la configuración de servicio:

```
[student@workstation ~]$ oc get -o yaml --export svc > /tmp/svc.yaml
```

- 6.2. Haga una copia del archivo exportado y limpie los atributos de tiempo de ejecución de este:

```
[student@workstation ~]$ cp /tmp/svc.yaml /tmp/svc-clean.yaml
```

- 6.3. Limpie los atributos de tiempo de ejecución del archivo /tmp/svc-clean.yaml.

Una copia del archivo limpiado está disponible en ~/D0288/labs/create-template/svc-clean.yaml. Compare el archivo limpio con esta versión y realice las ediciones necesarias para que sean iguales.

- 6.4. Elimine las primeras dos líneas del archivo:

```
apiVersion: v1
items:
```

- 6.5. Elimine todas las referencias a los atributos openshift.io/generated-by, creationTimestamp, namespace, resourceVersion, selfLink y uid del archivo.
- 6.6. Elimine el atributo managedFields y todos sus atributos secundarios.
- 6.7. Elimine todas las referencias al atributo clusterIP debajo del atributo spec del archivo.
- 6.8. Elimine el atributo status y todos sus atributos secundarios para ambos recursos de servicio.
- 6.9. Elimine el atributo kind: List al final del archivo y todos sus atributos secundarios.

► 7. Exporte y limpie la configuración de ruta para la aplicación quotesapi.

- 7.1. Exporte la configuración de ruta:

```
[student@workstation ~]$ oc get -o yaml --export route > /tmp/route.yaml
```

- 7.2. Haga una copia del archivo exportado y limpie los atributos de tiempo de ejecución de este:

```
[student@workstation ~]$ cp /tmp/route.yaml /tmp/route-clean.yaml
```

- 7.3. Limpie los atributos de tiempo de ejecución del archivo /tmp/route-clean.yaml.

Una copia del archivo limpiado está disponible en ~/D0288/labs/create-template/route-clean.yaml. Compare el archivo limpio con esta versión y realice las ediciones necesarias para que sean iguales.

- 7.4. Elimine las primeras dos líneas del archivo:

```
apiVersion: v1
items:
```

- 7.5. Elimine todas las referencias a los atributos openshift.io/generated-by, creationTimestamp, namespace, resourceVersion, selfLink y uid del archivo.

- 7.6. Elimine el atributo managedFields y todos sus atributos secundarios.

- 7.7. Elimine los atributos `host` y `subdomain` debajo del atributo `spec`.
- 7.8. Elimine el atributo `status` y todos sus atributos secundarios.
- 7.9. Elimine el atributo `kind: List` al final del archivo y todos sus atributos secundarios.
- 8. Exporte y limpie la configuración de volumen persistente (PVC) para la base de datos `quotesdb`.
- 8.1. Exporte la configuración de volumen persistente:
- ```
[student@workstation ~]$ oc get -o yaml --export pvc > /tmp/pvc.yaml
```
- 8.2. Haga una copia del archivo exportado y limpie los atributos de tiempo de ejecución de este:
- ```
[student@workstation ~]$ cp /tmp/pvc.yaml /tmp/pvc-clean.yaml
```
- 8.3. Limpie los atributos de tiempo de ejecución del archivo `/tmp/pvc-clean.yaml`. Una copia del archivo limpiado está disponible en `~/D0288/labs/create-template/pvc-clean.yaml`. Compare el archivo limpio con esta versión y realice las ediciones necesarias para que sean iguales.
- 8.4. Elimine las primeras dos líneas del archivo:
- ```
apiVersion: v1
items:
```
- 8.5. Elimine todos los atributos debajo del atributo `metadata.annotations`.
- 8.6. Elimine todas las referencias a `creationTimestamp`, el atributo `finalizers` y sus elementos secundarios, así como los atributos `namespace`, `resourceVersion`, `selfLink` y `uid` del archivo.
- 8.7. Elimine el atributo `managedFields` y todos sus atributos secundarios.
- 8.8. Elimine el atributo `dataSource` debajo del atributo `spec`.
- 8.9. Elimine los atributos `storageClassName`, `volumeMode` y `volumeName` debajo del atributo `spec`.
- 8.10. Elimine el atributo `status` y todos sus atributos secundarios.
- 8.11. Elimine el atributo `kind: List` al final del archivo y todos sus atributos secundarios.
- 9. Copie los fragmentos de código YAML de configuración de recursos individuales de los archivos limpios en el archivo `/home/student/quotes-template-clean.yaml` bajo el atributo `objects` en el siguiente orden:
- ```
[student@workstation ~]$ cat /tmp/is-clean.yaml >> ~/quotes-template-clean.yaml
[student@workstation ~]$ cat /tmp/bc-clean.yaml >> ~/quotes-template-clean.yaml
[student@workstation ~]$ cat /tmp/dc-clean.yaml >> ~/quotes-template-clean.yaml
[student@workstation ~]$ cat /tmp/svc-clean.yaml >> ~/quotes-template-clean.yaml
[student@workstation ~]$ cat /tmp/route-clean.yaml >> ~/quotes-template-clean.yaml
[student@workstation ~]$ cat /tmp/pvc-clean.yaml >> ~/quotes-template-clean.yaml
```

Hay una copia de la plantilla limpia disponible en `~/D0288/solutions/create-template/quotes-template-clean.yaml`. Compare la plantilla limpia final con este archivo.

Ignore las diferencias en los atributos `secret` y `uri`. Estos valores serán diferentes para usted. Parametrizará estos valores en el paso siguiente.

► 10. Agregue parámetros para poder volver a usar la plantilla.

Para mantener breve este ejercicio guiado, solo agregará tres parámetros: uno para la URL de Git donde se aloja el código fuente de la aplicación y dos para contraseñas y secretos.

- 10.1. Inspeccione el archivo que contiene la definición de parámetros. Observe que solo se requiere el parámetro `APP_GIT_URL`. Los parámetros `PASSWORD` y `SECRET` tienen valores predeterminados aleatorios:

```
[student@workstation ~]$ cat ~/D0288/labs/create-template/parameters.yaml
parameters:
- name: APP_GIT_URL
 displayName: Application Source Git URL
 description: The Git URL of the application source code
 required: true
- name: PASSWORD
 displayName: Database Password
 description: Password to access the database
 generate: expression
 from: '[a-zA-Z0-9]{16}'
- name: SECRET
 displayName: Webhook Secret
 description: Secret for webhooks
 generate: expression
 from: '[a-zA-Z0-9]{40}'
```

- 10.2. Haga una copia del archivo YAML de la plantilla limpia final antes de agregar los parámetros:

```
[student@workstation ~]$ cp ~/quotes-template-clean.yaml ~/quotes-template.yaml
```

- 10.3. Abra el archivo `quotes-template.yaml` en un editor de texto. Copie el contenido del archivo `parameters.yaml` al final del archivo `quotes-template.yaml`.

- 10.4. Cambie los atributos `secret`, `password` y `uri` para hacer referencia a los parámetros de la plantilla.

Use la siguiente lista como guía para realizar estos cambios:

```
...output omitted...
 kind: BuildConfig
 ...output omitted...
 name: quotesapi
 ...output omitted...
 source:
 contextDir: quotes
 git:
 uri: ${APP_GIT_URL}
```

```

type: Git
...output omitted...
triggers:
- github:
 secret: ${SECRET}
type: GitHub
- generic:
 secret: ${SECRET}
...output omitted...
kind: DeploymentConfig
...output omitted...
name: quotesapi
...output omitted...
- name: DATABASE_PASSWORD
 value: ${PASSWORD}
...output omitted...
kind: DeploymentConfig
...output omitted...
name: quotesdb
...output omitted...
- name: MYSQL_PASSWORD
 value: ${PASSWORD}
...output omitted...

```

- 10.5. Para que el contenedor de base de datos MySQL sea persistente, debe agregar un atributo `volumeMounts` a la configuración de implementación de la base de datos y hacer referencia al recurso de reclamación de volumen persistente declarado en la plantilla.

Agregue las siguientes líneas a la configuración de implementación de quotesdb de la siguiente manera:

```

...output omitted...
terminationMessagePath: /dev/termination-log
terminationMessagePolicy: File
volumeMounts:
- mountPath: /var/lib/mysql/data
 name: quotesdb-volume-1
dnsPolicy: ClusterFirst
restartPolicy: Always
...output omitted...

```

- 10.6. El archivo de la plantilla ahora está completo. Guarde sus ediciones.

Para verificar los cambios que realizó durante este paso, revise el archivo `quotes-template.yaml` en la carpeta `~/DO288/solutions/create-template`. Si no está seguro acerca de sus ediciones, puede copiar el archivo de soluciones y continuar con el siguiente paso.

- 11. Cree una aplicación nueva desde una plantilla.

- 11.1. Cree el proyecto `youruser-myquotes`:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-myquotes
Now using project "youruser-myquotes" on server
"https://api.cluster.domain.example.com:6443"
...output omitted...
```

- 12. Cree una nueva instancia de la aplicación quotes desde una plantilla.

- 12.1. Use el comando `oc new-app` para crear una nueva instancia de la aplicación quotes desde la plantilla:

```
[student@workstation ~]$ oc new-app --file=quotes-template.yaml \
> -p APP_GIT_URL=https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
> -p PASSWORD=mypass
--> Deploying template "youruser-myquotes/quotes" to project youruser-myquotes

Quotes Application

The Quotes application provides an HTTP API that returns a random, funny quote.

* With parameters:
 * Application Source Git URL=https://github.com/youruser/D0288-apps
 * Database Password=mypass
 * Webhook Secret=D6xlih2F3KIyYwsIXs3nLysGHJbi6JLhtaul6I10 # generated

--> Creating resources ...
imagestream.image.openshift.io "quotesapi" created
buildconfig.build.openshift.io "quotesapi" created
deploymentconfig.apps.openshift.io "quotesapi" created
deploymentconfig.apps.openshift.io "quotesdb" created
service "quotesapi" created
service "quotesdb" created
route.route.openshift.io "quotesapi" created
persistentvolumeclaim "quotesdb-claim" created
--> Success
...output omitted...
```



#### nota

Si ve un error en este paso, asegúrese de que no haya saltos de línea adicionales y de que el archivo YAML de plantilla tenga una sangría adecuada. OpenShift muestra el número de línea en la plantilla donde se produjo un error en el procesamiento. Use esta información para identificar y solucionar los problemas.

- 12.2. Espere a que la compilación se complete y las aplicaciones quotesdb y quotesapi tengan cada una un pod listo y en ejecución.

```
[student@workstation ~]$ oc get pods
NAME READY STATUS RESTARTS AGE
...output omitted...
quotesapi-1-tphzw 1/1 Running 0 3m12s
quotesdb-1-rk8mw 1/1 Running 0 4m28s
```

## ► 13. Complete la base de datos y pruebe la aplicación.

## 13.1. Inspecione el script que completa la base de datos.

El script `populate-db.sh` del directorio `~/D0288/labs/create-template` usa un túnel de redireccionamiento de puertos para conectarse al pod de la base de datos y un cliente MySQL local para ejecutar un script de SQL:

```
[student@workstation ~]$ cat ~/D0288/labs/create-template/populate-db.sh
...output omitted...
echo 'Creating tunnel...'
pod=$(oc get pod -l deploymentconfig=quotesdb -o name)
oc port-forward $(basename ${pod}) 30306:3306 &
tunnel=$!
sleep 3
echo 'Initializing the database...'
mysql -h127.0.0.1 -P30306 -uquoteapp -pmypass quotesdb \
< ~/D0288/labs/create-template/quote.sql
sleep 3
echo 'Terminating tunnel...'
kill ${tunnel}
```

13.2. Ejecute el script `populate-db.sh`:

```
[student@workstation ~]$ ~/D0288/labs/create-template/populate-db.sh
Creating tunnel...
Forwarding from 127.0.0.1:30306 -> 3306
Forwarding from [::1]:30306 -> 3306
Initializing the database...
Handling connection for 30306
Terminating tunnel...
```

## 13.3. Obtenga el nombre de host de ruta para la aplicación:

```
[student@workstation ~]$ oc get route/quotesapi -o jsonpath='{.spec.host}{"\n"}'
quotesapi-youruser-quotes-dev.apps.cluster.domain.example.com
```

13.4. Use el comando `curl` y el nombre de host del paso anterior para acceder a la aplicación. Su salida puede ser diferente a este ejemplo:

```
[student@workstation ~]$ curl \
> quotesapi-${RHT_OCP4_DEV_USER}-myquotes.${RHT_OCP4_WILDCARD_DOMAIN}/get.php
Veni, vidi, vici...
```

## ► 14. Realice la limpieza. Elimine los proyectos de OpenShift y lance el almacenamiento persistente asignado para este ejercicio.

Abra una ventana de terminal en la máquina virtual `workstation` y ejecute el siguiente comando para realizar las tareas de limpieza:

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

Esto concluye el ejercicio guiado.

## ► Trabajo de laboratorio

# Creación de aplicaciones desde plantillas de OpenShift

### Lista de verificación de rendimiento

En este trabajo de laboratorio, implementará una aplicación To Do List de una plantilla que completará para su reutilización.

### Resultados

También deberá ser capaz de completar una plantilla que implementa un pod de base de datos y un pod de aplicación web, usar la plantilla para implementar la aplicación y verificar que la misma funcione.

### Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación de muestra en el repositorio Git (todo-single).
- Las dependencias npm requeridas por la aplicación (`restify`, `sequelize` y `mysql`).
- La imagen del compilador S2I Node.js 10 y la imagen de la base de datos MySQL 5.7 requeridas por la plantilla.

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de proyectos de inicio y de soluciones:

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

### Requisitos

La aplicación es una aplicación To Do List escrita en JavaScript. Consiste de un front-end web, basado en el marco (framework) web AngularJS, y un back-end API HTTP, basado en Node.js. El back-end usa los marcos (frameworks) Restify y Sequelize.

Tanto el front-end como el back-end de la aplicación se ejecutan en el mismo contenedor. La aplicación se implementa de un código fuente almacenado en un repositorio Git. Una base de datos MySQL se usa como almacén de datos. La aplicación inicializa la base de datos durante el inicio.

La plantilla toma los siguientes parámetros:

- `APP_GIT_URL`: la URL de Git donde se almacena el código fuente de la aplicación To Do List.
- `HOSTNAME`: el nombre del host usado para acceder a la aplicación To Do List.
- `NPM_PROXY`: la URL del servidor del repositorio NPM.

- SECRET: el secreto para webhooks de OpenShift.
- PASSWORD: la contraseña de conexión de la base de datos. El nombre de usuario se fija en la plantilla.
- CLEAN\_DATABASE: un marcador que indica si la aplicación inicializa la base de datos durante el inicio.

Los parámetros de la plantilla tienen las siguientes restricciones:

- Los parámetros APP\_GIT\_URL, HOSTNAME, NPM\_PROXY y CLEAN\_DATABASE son obligatorios.
- Los parámetros SECRET y PASSWORD tienen valores predeterminados aleatorios.
- El parámetro CLEAN\_DATABASE tiene el valor predeterminado de `true`.

Un archivo de plantilla de inicio denominado `todo-template.yaml` se ofrece en la carpeta `~/D0288/labs/review-template`. Contiene los recursos requeridos para implementar la aplicación. Los recursos se encuentran en el orden correcto y ya están limpios. Se requiere que agregue los parámetros que faltan y agregue referencias a los parámetros en la lista de recursos.

Implemente la aplicación según los siguientes requisitos:

- El proyecto de la aplicación se denomina `youruser-review-template`.
- Use el script `~/D0288/labs/review-template/oc-new-app.sh` para implementar la aplicación. Edite el script según sea necesario para pasar los parámetros requeridos.
- La aplicación inicializa la base de datos durante el inicio.
- La contraseña para acceder a la base de datos es `mypass`.
- Los módulos Npm requeridos para compilar la aplicación están disponibles en:

`http://nexus-common.apps.cluster.domain.example.com/repository/nodejs`

- Si prefiere probar la aplicación con un navegador web, use el front-end web para agregar algunas entradas a la aplicación To Do List de manera interactiva. La interfaz de usuario de la aplicación es accesible a través de la siguiente URL:

`http://youruser-todo.apps.cluster.domain.example.com/todo/index.html`

- Si prefiere probar la aplicación usando la línea de comando, use la siguiente URL para enviar una solicitud HTTP GET al back-end de la aplicación:

`http://youruser-todo.apps.cluster.domain.example.com/todo/api/items-count`

La respuesta incluye un atributo `count`, inclusive si no hay entradas en la aplicación. Una respuesta de error implica que la base de datos no se inicializó correctamente.

## Pasos

1. Revise el archivo de la plantilla de inicio `todo-template.yaml` en la carpeta `~/D0288/labs/review-template`. Haga una copia de este archivo en el directorio `/home/student/` y use esta copia para realizar los cambios en todo el trabajo de laboratorio. Identifique y agregue cualquier parámetro que falte al final del archivo. Identifique las referencias de parámetros que faltan de la lista de recursos de plantilla y agréguelas según

sea necesario. Use las definiciones de parámetros y las referencias de parámetros existentes como guía para agregar las que faltan.

No es necesario, pero puede inspeccionar el archivo `todo-template-clean.yaml` en la misma carpeta. Este archivo contiene los recursos de aplicación exportados por el comando `oc get`, con los atributos de tiempo de ejecución ya eliminados. No realice ningún cambio a este archivo.

No se requiere que lo haga, pero también puede inspeccionar los archivos `new-app-db.sh`, `new-app-node.sh` y `add-route.sh` en la misma carpeta, los cuales contienen los comandos usados para generar los recursos exportados al archivo `todo-template-clean.yaml`. No realice cambios a estos archivos y no ejecute ninguno.

2. Cree un nuevo proyecto e implemente la aplicación To Do List usando el archivo de definición de la plantilla que completó durante el paso anterior.

Revise el archivo del script de inicio `oc-new-app.sh` en la carpeta `~/D0288/labs/review-template`. Haga una copia de este archivo en el directorio `/home/student/` y use esta copia para realizar los cambios en todo el trabajo de laboratorio. Identifique y agregue cualquier parámetro que falte a la línea de comando `oc new-app`. Ejecute el script `oc-new-app.sh` finalizado para implementar la aplicación.

No realice ningún paso manual para inicializar la base de datos. La aplicación To Do List crea las tablas de la base de datos requerida si usted pasa el conjunto correcto de parámetros a la plantilla. Las tablas de la base de datos no necesitan un conjunto de datos inicial.

3. Pruebe la aplicación To Do List usando un navegador web o la línea de comando. No olvide obtener las variables de `/usr/local/etc/ocp4.config` antes de iniciar sesión en OpenShift.
4. Califique su trabajo.

Ejecute el siguiente comando en la máquina virtual `workstation` para verificar que se hayan realizado todas las tareas:

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

5. Realice la limpieza. Elimine el proyecto de OpenShift.

Abra una ventana de terminal en la máquina virtual `workstation` y ejecute el siguiente comando para realizar las tareas de limpieza:

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

Esto concluye el trabajo de laboratorio.

## ► Solución

# Creación de aplicaciones desde plantillas de OpenShift

### Lista de verificación de rendimiento

En este trabajo de laboratorio, implementará una aplicación To Do List de una plantilla que completará para su reutilización.

### Resultados

También deberá ser capaz de completar una plantilla que implementa un pod de base de datos y un pod de aplicación web, usar la plantilla para implementar la aplicación y verificar que la misma funcione.

### Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación de muestra en el repositorio Git (todo-single).
- Las dependencias npm requeridas por la aplicación (`restify`, `sequelize` y `mysql`).
- La imagen del compilador S2I Node.js 10 y la imagen de la base de datos MySQL 5.7 requeridas por la plantilla.

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de proyectos de inicio y de soluciones:

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

### Requisitos

La aplicación es una aplicación To Do List escrita en JavaScript. Consiste de un front-end web, basado en el marco (framework) web AngularJS, y un back-end API HTTP, basado en Node.js. El back-end usa los marcos (frameworks) Restify y Sequelize.

Tanto el front-end como el back-end de la aplicación se ejecutan en el mismo contenedor. La aplicación se implementa de un código fuente almacenado en un repositorio Git. Una base de datos MySQL se usa como almacén de datos. La aplicación inicializa la base de datos durante el inicio.

La plantilla toma los siguientes parámetros:

- `APP_GIT_URL`: la URL de Git donde se almacena el código fuente de la aplicación To Do List.
- `HOSTNAME`: el nombre del host usado para acceder a la aplicación To Do List.
- `NPM_PROXY`: la URL del servidor del repositorio NPM.
- `SECRET`: el secreto para webhooks de OpenShift.

- **PASSWORD:** la contraseña de conexión de la base de datos. El nombre de usuario se fija en la plantilla.
- **CLEAN\_DATABASE:** un marcador que indica si la aplicación inicializa la base de datos durante el inicio.

Los parámetros de la plantilla tienen las siguientes restricciones:

- Los parámetros APP\_GIT\_URL, HOSTNAME, NPM\_PROXY y CLEAN\_DATABASE son obligatorios.
- Los parámetros SECRET y PASSWORD tienen valores predeterminados aleatorios.
- El parámetro CLEAN\_DATABASE tiene el valor predeterminado de true.

Un archivo de plantilla de inicio denominado `todo-template.yaml` se ofrece en la carpeta `~/D0288/labs/review-template`. Contiene los recursos requeridos para implementar la aplicación. Los recursos se encuentran en el orden correcto y ya están limpios. Se requiere que agregue los parámetros que faltan y agregue referencias a los parámetros en la lista de recursos.

Implemente la aplicación según los siguientes requisitos:

- El proyecto de la aplicación se denomina `youruser-review-template`.
- Use el script `~/D0288/labs/review-template/oc-new-app.sh` para implementar la aplicación. Edite el script según sea necesario para pasar los parámetros requeridos.
- La aplicación inicializa la base de datos durante el inicio.
- La contraseña para acceder a la base de datos es `mypass`.
- Los módulos Npm requeridos para compilar la aplicación están disponibles en:  
`http://nexus-common.apps.cluster.domain.example.com/repository/nodejs`
- Si prefiere probar la aplicación con un navegador web, use el front-end web para agregar algunas entradas a la aplicación To Do List de manera interactiva. La interfaz de usuario de la aplicación es accesible a través de la siguiente URL:  
`http://youruser-todo.apps.cluster.domain.example.com/todo/index.html`
- Si prefiere probar la aplicación usando la línea de comando, use la siguiente URL para enviar una solicitud HTTP GET al back-end de la aplicación:  
`http://youruser-todo.apps.cluster.domain.example.com/todo/api/items-count`

La respuesta incluye un atributo `count`, inclusive si no hay entradas en la aplicación. Una respuesta de error implica que la base de datos no se inicializó correctamente.

## Pasos

1. Revise el archivo de la plantilla de inicio `todo-template.yaml` en la carpeta `~/D0288/labs/review-template`. Haga una copia de este archivo en el directorio `/home/student/` y use esta copia para realizar los cambios en todo el trabajo de laboratorio. Identifique y agregue cualquier parámetro que falte al final del archivo. Identifique las referencias de parámetros que faltan de la lista de recursos de plantilla y agréguelas según sea necesario. Use las definiciones de parámetros y las referencias de parámetros existentes como guía para agregar las que faltan.

No es necesario, pero puede inspeccionar el archivo `todo-template-clean.yaml` en la misma carpeta. Este archivo contiene los recursos de aplicación exportados por el comando `oc get`, con los atributos de tiempo de ejecución ya eliminados. No realice ningún cambio a este archivo.

No se requiere que lo haga, pero también puede inspeccionar los archivos `new-app-db.sh`, `new-app-node.sh` y `add-route.sh` en la misma carpeta, los cuales contienen los comandos usados para generar los recursos exportados al archivo `todo-template-clean.yaml`. No realice cambios a estos archivos y no ejecute ninguno.

- 1.1. Cree una copia de la plantilla de inicio para realizar ediciones:

```
[student@workstation ~]$ cp ~/D0288/labs/review-template/todo-template.yaml \
> ~/todo-template.yaml
```

- 1.2. Identifique cualquier parámetro que falta en el archivo de la plantilla.

Abra el archivo `~/todo-template.yaml` con un editor de texto.

Desplácese hacia el final del archivo. El archivo contiene cuatro parámetros denominados `APP_GIT_URL`, `HOSTNAME`, `NPM_PROXY` y `SECRET`. Estas definiciones de parámetros están completas y no necesitan ningún tipo de modificación.

Faltan los parámetros `PASSWORD` y `CLEAN_DATABASE`, y se los debe agregar.

- 1.3. Agregue los parámetros que faltan al archivo de la plantilla `~/todo-template.yaml`.

Agregue las siguientes líneas al archivo. Puede copiar estas líneas desde el archivo `add-parameters.yaml` en la carpeta `~/D0288/solutions/review-template`:

```
- name: PASSWORD
 displayName: Database Password
 description: Password to access the database
 generate: expression
 from: '[a-zA-Z0-9]{16}'
- name: CLEAN_DATABASE
 displayName: Initialize the database
 description: If 'true', the database is cleaned when the application starts.
 required: true
 value: "true"
```

- 1.4. Identifique las referencias que faltan en los valores de parámetros de la lista de recursos de la plantilla.

Algunos de los recursos ya hacen referencia a los parámetros correctamente. La siguiente lista destaca las referencias de parámetros que ya existen en el archivo de la plantilla. No necesita cambiar ninguna:

```
...output omitted...
kind: BuildConfig
...output omitted...
 name: todoapp
 ...output omitted...
 triggers:
 - github:
 secret: ${SECRET}
 type: GitHub
 - generic:
```

```

secret: ${SECRET}
...output omitted...
kind: DeploymentConfig
...output omitted...
name: todoapp
...output omitted...
- env:
 - name: DATABASE_NAME
 value: tododb
 - name: DATABASE_PASSWORD
 value: ${PASSWORD}
 - name: DATABASE_SVC
 value: tododb
 - name: DATABASE_USER
 value: todoapp
 - name: DATABASE_INIT
 value: ${CLEAN_DATABASE}
...output omitted...

```

En algunos recursos, faltan las referencias a parámetros. La siguiente lista destaca los atributos que se establecen como valores codificados y se deben reemplazar por referencias de parámetros:

```

...output omitted...
kind: BuildConfig
...output omitted...
name: todoapp
...output omitted...
strategy:
 sourceStrategy:
 env:
 - name: npm_config_registry
 value: http://INVALIDHOST.NODOMAIN.NUL
...output omitted...
kind: DeploymentConfig
...output omitted...
name: tododb
...output omitted...
- env:
 - name: MYSQL_DATABASE
 value: tododb
 - name: MYSQL_PASSWORD
 value: FIXEDPASSWD
...output omitted...
kind: Route
...output omitted...
name: todoapp
...output omitted...
spec:
 host: http://INVALIDHOST.NODOMAIN.NUL
...output omitted...

```

- 1.5. Agregue las referencias que faltan a los parámetros al archivo ~/todo-template.yaml. En la siguiente lista, se muestran los cambios que debe realizar:

```

...output omitted...
kind: BuildConfig
...output omitted...
name: todoapp
...output omitted...
strategy:
 sourceStrategy:
 env:
 - name: npm_config_registry
 value: ${NPM_PROXY}
...output omitted...
kind: DeploymentConfig
...output omitted...
name: tododb
...output omitted...
 - env:
 - name: MYSQL_DATABASE
 value: tododb
 - name: MYSQL_PASSWORD
 value: ${PASSWORD}
...output omitted...
kind: Route
...output omitted...
name: todoapp
...output omitted...
spec:
 host: ${HOSTNAME}
...output omitted...

```

- 1.6. Revise sus ediciones y guarde los cambios en el archivo de la plantilla ~/todo-template.yaml. Puede comparar sus ediciones con el archivo de solución en la carpeta ~/D0288/solutions/review-template.
  2. Cree un nuevo proyecto e implemente la aplicación To Do List usando el archivo de definición de la plantilla que completó durante el paso anterior.  
Revise el archivo del script de inicio oc-new-app.sh en la carpeta ~/D0288/labs/review-template. Haga una copia de este archivo en el directorio /home/student/ y use esta copia para realizar los cambios en todo el trabajo de laboratorio. Identifique y agregue cualquier parámetro que falte a la línea de comando oc new-app. Ejecute el script oc-new-app.sh finalizado para implementar la aplicación.  
No realice ningún paso manual para inicializar la base de datos. La aplicación To Do List crea las tablas de la base de datos requerida si usted pasa el conjunto correcto de parámetros a la plantilla. Las tablas de la base de datos no necesitan un conjunto de datos inicial.
- 2.1. Cree una copia del script de inicio para realizar ediciones:
- ```
[student@workstation ~]$ cp ~/D0288/labs/review-template/oc-new-app.sh \
> ~/oc-new-app.sh
```
- 2.2. Identifique cualquier parámetro que falta en el archivo del script ~/oc-new-app.sh.
Abra el archivo ~/oc-new-app.sh con un editor de texto.

El script pasa valores para los parámetros APP_GIT_URL, NPM_PROXY, PASSWORD y CLEAN_DATABASE.

Los valores pasados para APP_GIT_URL, NPM_PROXY y PASSWORD son correctos; no necesita modificarlos. Debe cambiar el valor para el parámetro CLEAN_DATABASE a true.

También debe agregar una opción de línea de comando para pasar un valor para el parámetro HOSTNAME.

2.3. Realice cambios al script ~/oc-new-app.sh.

Use la siguiente lista como guía para realizar estos cambios:

```
oc new-app --as-deployment-config --name todo --file ~/todo-template.yaml \
> -p APP_GIT_URL=https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
> -p NPM_PROXY=http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs \
> -p PASSWORD=mypass \
> -p CLEAN_DATABASE=true \
> -p HOSTNAME=${RHT_OCP4_DEV_USER}-todo.${RHT_OCP4_WILDCARD_DOMAIN}
```

No agregue saltos de línea a ningún valor de parámetro. El valor para NPM_PROXY debe estar en una única línea. Recuerde que no necesita cambiarlo del archivo de inicio.

2.4. Revise sus ediciones y guarde los cambios en el archivo ~/oc-new-app.sh.

Puede comparar sus ediciones con el archivo de solución en la carpeta ~/D0288/solutions/review-template.

3. Pruebe la aplicación To Do List usando un navegador web o la línea de comando. No olvide obtener las variables de /usr/local/etc/ocp4.config antes de iniciar sesión en OpenShift.

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 ~]$ source /usr/local/etc/ocp4.config
```

3.2. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

3.3. Cree el proyecto youruser-review-template:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-review-template
Now using project "youruser-review-template" on server
"https://api.cluster.domain.example.com:6443"
...output omitted...
```

3.4. Ejecute el script oc-new-app.sh para crear todos los recursos del archivo de la plantilla todo-template.yaml:

```
[student@workstation ~]$ ~/oc-new-app.sh
...output omitted...
--> Creating resources...
  imagestream "todoapp" created
  buildconfig "todoapp" created
  deploymentconfig "todoapp" created
  deploymentconfig "tododb" created
  service "todoapp" created
  service "tododb" created
  route "todoapp" created
--> Success
...output omitted...
```

3.5. Espere que finalice la compilación de la aplicación To Do List:

```
[student@workstation ~]$ oc logs bc/todoapp -f
...output omitted...
Push successful
```

3.6. Espere a que la aplicación y el pod de la base de datos estén listos y en ejecución:

| NAME | READY | STATUS | RESTARTS | AGE |
|----------------------|-------|---------|----------|-----|
| ...output omitted... | | | | |
| todoapp-1-nch6c | 1/1 | Running | 0 | 1m |
| tododb-1-lpk63 | 1/1 | Running | 0 | 2m |

3.7. Obtenga el nombre de host de ruta para la aplicación:

```
[student@workstation ~]$ oc get route/todoapp -o jsonpath='{.spec.host}{"\n"}'
youruser-todo.apps.cluster.domain.example.com
```

3.8. Pruebe la aplicación con el nombre de host de ruta del paso anterior.

Si prefiere probar la aplicación To Do List usando un navegador web, acceda a la siguiente URL y agregue algunas entradas a la lista:

<http://youruser-todo.apps.cluster.domain.example.com/todo/index.html>

3.9. Si prefiere probar la aplicación To Do List usando la línea de comando, abra una ventana de terminal y ejecute el siguiente comando:

```
[student@workstation ~]$ curl -siw "\n" \
> http://${RHT_OCP4_DEV_USER}-todo.${RHT_OCP4_WILDCARD_DOMAIN}\
> /todo/api/items-count
HTTP/1.1 200 OK
...output omitted...
{"count":0}
```

4. Califique su trabajo.

Ejecute el siguiente comando en la máquina virtual **workstation** para verificar que se hayan realizado todas las tareas:

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

5. Realice la limpieza. Elimine el proyecto de OpenShift.

Abra una ventana de terminal en la máquina virtual **workstation** y ejecute el siguiente comando para realizar las tareas de limpieza:

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Una plantilla es una lista con parámetros de recursos de OpenShift. Un caso de uso típico para plantillas es implementar aplicaciones compuestas por varios pods.
- Los parámetros de la plantilla pueden ser requeridos u opcionales y pueden definir valores predeterminados generados en forma aleatoria, los cuales son útiles para secretos y contraseñas.
- Los típicos métodos para crear una plantilla son la concatenación de la salida de varios comandos `oc new-app -o` y la exportación de recursos existentes mediante el comando `oc get -o yaml --export`.
- Los comandos `oc describe` y `oc explain` ayudan a los desarrolladores a encontrar información acerca de todos los atributos válidos para un tipo de recurso dado.

capítulo 7

Administración de las implementaciones de aplicaciones

Meta

Monitorear el estado de la aplicación y usar diferentes métodos de implementación para aplicaciones nativas de la nube.

Objetivos

- Implementar sondeos de disponibilidad y ejecución.
- Seleccionar la estrategia de implementación adecuada para una aplicación nativa de la nube.
- Administrar la implementación de una aplicación con comandos de CLI.

Secciones

- Monitoreo del estado de la aplicación (y ejercicio guiado)
- Selección de una estrategia de implementación adecuada (y ejercicio guiado)
- Administración de implementaciones de aplicaciones con comandos de CLI (y ejercicio guiado)

Trabajo de laboratorio

Administración de las implementaciones de aplicaciones

Monitoreo del estado de la aplicación

Objetivos

Después de completar esta sección, deberá ser capaz de implementar los sondeos de disponibilidad y ejecución para monitorear la disponibilidad y el estado de la aplicación.

Sondeos de disponibilidad y ejecución de OpenShift

Las aplicaciones pueden tener un estado poco fiable por varios motivos, que incluyen pérdida temporal de conectividad, errores de configuración o errores de aplicación. Los desarrolladores pueden usar *sondeos* para monitorear sus aplicaciones y tener en cuenta los eventos que pueden variar del estado al uso de recursos y errores. Este monitoreo es útil para solucionar problemas, pero también puede ayudar con la planificación y administración de recursos. Un sondeo es una acción de OpenShift que realiza diagnósticos periódicamente en un contenedor en ejecución. Los sondeos se pueden configurar con el cliente de la línea de comandos oc, definido como parte de la plantilla de OpenShift o mediante la consola web de OpenShift. Actualmente, existen dos tipos de sondeos en OpenShift:

Sondeo de disponibilidad

Los sondeos de disponibilidad determinan si un contenedor está listo o no para responder las solicitudes. Si el sondeo de disponibilidad devuelve un estado de error, OpenShift quita la dirección IP del contenedor de los extremos (endpoints) de todos los servicios.

Los desarrolladores pueden usar sondeos de disponibilidad para indicar a OpenShift que, aunque se está ejecutando un contenedor, no debe recibir tráfico de un proxy. El sondeo de disponibilidad está configurado en el atributo spec.spec.containers.readinessprobe de la configuración de pod.

Sondeo de ejecución

Los sondeos de ejecución determinan si una aplicación que se está ejecutando en un contenedor tiene un estado healthy (correcto). Si el sondeo de ejecución detecta un estado incorrecto, OpenShift desactiva el contenedor e intenta volver a implementarlo. El sondeo de ejecución está configurado en el atributo spec.spec.containers.livenessprobe de la configuración de pod.

OpenShift ofrece cinco opciones que controlan estos dos sondeos:

| Nombre | Obligatorio | Descripción | Valor predeterminado |
|---------------------|-------------|---|----------------------|
| initialDelaySeconds | Sí | Determina cuánto tiempo se debe esperar después de que se inicia el contenedor y antes de comenzar el sondeo. | 0 |

| Nombre | Obligatorio | Descripción | Valor predeterminado |
|------------------|-------------|--|----------------------|
| timeoutSeconds | Sí | Determina cuánto tiempo se debe esperar para que finalice el sondeo. Si se excede este tiempo, OpenShift asume que el sondeo falló. | 1 |
| periodSeconds | No | Especifica la frecuencia de las comprobaciones. | 1 |
| successThreshold | No | Especifica los mínimos aciertos consecutivos del sondeo para que se considere satisfactorio después de haber fallado. | 1 |
| failureThreshold | No | Especifica los mínimos errores consecutivos del sondeo para que se considere con errores después de haber tenido un resultado satisfactorio. | 3 |

Métodos para comprobar el estado de la aplicación

Los sondeos de disponibilidad y ejecución pueden comprobar el estado de las aplicaciones de tres maneras:

Comprobaciones HTTP

Una comprobación HTTP es ideal para aplicaciones que muestran códigos de estado HTTP, como las API REST.

OpenShift usa solicitudes HTTP GET para verificar el código de estado de respuestas para determinar el estado de un contenedor. La verificación se considera exitosa si el código de respuesta HTTP es entre 200 y 399. En el siguiente ejemplo, se demuestra cómo implementar un sondeo de disponibilidad con el método de comprobación HTTP:

```
...
readinessProbe:
  httpGet:
    path: /health❶
    port: 8080
  initialDelaySeconds: 15❷
  timeoutSeconds: 1❸
...
```

- ❶ La URL que se consultará.
- ❷ Cuánto tiempo se debe esperar después de que se inicia el contenedor antes de comprobar su estado.
- ❸ Cuánto tiempo se debe esperar para que finalice el sondeo.

Comprobaciones de ejecución de contenedores

Las comprobaciones de ejecución del contenedor son ideales en escenarios en los que debe determinar el estado de disponibilidad y ejecución del contenedor según el código de salida de un proceso o script de shell que se ejecuta en el contenedor.

Cuando se usan comprobaciones de ejecución de contenedores, OpenShift ejecuta un comando dentro del contenedor. Salir de la comprobación con un estado de 0 se considera un resultado satisfactorio. Todos los demás estados se consideran un error. En el siguiente ejemplo, se demuestra cómo implementar una comprobación de ejecución de contenedores:

```
...
livenessProbe:
  exec:
    command: ①
      - cat
      - /tmp/health
  initialDelaySeconds: 15
  timeoutSeconds: 1
...
```

- ① El comando a ejecutar y sus argumentos, como una matriz YAML.

Comprobaciones de sockets TCP

Una comprobación de sockets TCP es ideal para aplicaciones que se ejecutan como daemons y puertos TCP abiertos, como servidores de bases de datos, servidores de archivos, servidores web y servidores de aplicaciones.

Cuando se usan las comprobaciones de sockets TCP, OpenShift intenta abrir un socket en el contenedor. Se considera que el contenedor tiene un estado correcto si la comprobación puede establecer una conexión exitosa. En el siguiente ejemplo, se demuestra cómo implementar un sondeo de ejecución con el método de comprobación de sockets TCP:

```
...
livenessProbe:
  tcpSocket:
    port: 8080①
  initialDelaySeconds: 15
  timeoutSeconds: 1
...
```

- ① El puerto TCP que se comprobará.

Administración de sondeos con la consola web

Los desarrolladores pueden crear sondeos de disponibilidad y ejecución mediante la edición del archivo YAML de configuración de implementación con `oc edit` o la consola web de OpenShift. Puede editar directamente el archivo YAML de configuración de implementación desde la página **Workloads (Cargas de trabajo) → Deployment Configs (Configuración de implementación)** → <nombre de la implementación> de la consola web. Haga clic en el menú desplegable **Actions (Acciones)** y seleccione **Edit Deployment Config (Editar configuración de implementación)**.

Deployment Configs > Deployment Config Details

dc hello

Deployment Config Details

1 pod

| | | | |
|-----------|--------------|----------------|---------------|
| Name | hello | Latest Version | 1 |
| Namespace | your-project | Message | config change |

Actions

- Start Rollout
- Pause Rollouts
- Edit Pod Count
- Add Storage
- Edit Health Checks
- Edit Labels
- Edit Annotations
- Edit Deployment Config**
- Delete Deployment Config

Figura 7.1: Adición de sondeos con la consola web

En el siguiente ejemplo, se muestra el editor YAML en la consola web para una configuración de implementación.

```

191      terminationMessagePath: /dev/termination-log
192      name: httpd-example
193      livenessProbe:
194        httpGet:
195          path: /healthz
196          port: 8080
197          scheme: HTTP
198          initialDelaySeconds: 30
199          timeoutSeconds: 3
200          periodSeconds: 10
201          successThreshold: 1
202          failureThreshold: 3
203        ports:
204          - containerPort: 8080
205            protocol: TCP
imagePullPolicy: IfNotPresent
  
```

Actions

View shortcuts

Details **YAML** Replication Controllers Pods Environment Events

Save Reload Cancel Download

Figura 7.2: Adición de sondeos con el editor YAML de la consola web

Creación de sondeos con CLI

El comando `oc set probe` proporciona un enfoque alternativo para editar la definición YAML de configuración de implementación directamente. Cuando se crean sondeos para aplicaciones en ejecución, el enfoque recomendado es usar `oc set probe`, ya que limita su capacidad de cometer errores. Este comando proporciona una serie de opciones que le permiten especificar el tipo de sondeo, ya sea de disponibilidad o de ejecución, así como otros atributos necesarios como puerto, URL, tiempo de espera, período y más.

Los siguientes ejemplos muestran el uso del comando `oc set probe` con diversas opciones:

```
[user@host ~]$ oc set probe dc/myapp --readiness \
> --get-url=http://:8080/healthz --period=20
```

```
[user@host ~]$ oc set probe dc/myapp --liveness \
> --open-tcp=3306 --period=20 \
> --timeout-seconds=1
```

```
[user@host ~]$ oc set probe dc/myapp --liveness \
> --get-url=http://:8080/healthz --initial-delay-seconds=30 \
> --success-threshold=1 --failure-threshold=3
```

Use el comando `oc set probe --help` para ver todas las opciones disponibles para este comando.



Referencias

Para obtener más información, consulte la sección *Monitoreo del estado de la aplicación* del capítulo *Aplicaciones* de la Documentación oficial para OpenShift Container Platform 4.5; en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/applications/application-health

Para obtener más información, consulte la página *Configurar sondeos de ejecución y disponibilidad* del sitio web de Kubernetes en <https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/>

► Ejercicio Guiado

Activación de sondeos

En este ejercicio, activará los sondeos de ejecución y disponibilidad para monitorear el estado de una aplicación implementada en un clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Activar los sondeos de disponibilidad y ejecución para un aplicación desde la línea de comandos.
- Buscar mensajes de error de sondeo en el registro de eventos.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Imagen de compilador S2I Node.js.
- La aplicación de muestra del repositorio Git (sondeos).

Ejecute el siguiente comando en `workstation` para validar los requisitos previos del ejercicio y descargar los archivos de trabajos de laboratorio:

```
[student@workstation ~]$ lab probes start
```

La aplicación que implemente en este ejercicio expone dos extremos (endpoints) GET de HTTP en `/healthz` y `/ready`. El extremo (endpoint) `/healthz` se usará para el sondeo de ejecución y el sondeo `/ready` se usará para el sondeo de disponibilidad. Los extremos (endpoints) responden con un código de estado HTTP de 200 si el pod está listo y en estado correcto; de lo contrario, el código de estado es 503.

► 1. Cree un nuevo proyecto y, luego, implemente la aplicación de muestra en el subdirectorio `probes` del repositorio Git en un clúster OpenShift.

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 OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
```

13. Cree un proyecto nuevo:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-probes
```

14. Cree una nueva aplicación denominada probes desde las fuentes del subdirectorio probes del repositorio Git.

Puede copiar o ejecutar el comando del script `oc-new-app.sh` en la carpeta `/home/student/D0288/labs/probes`:

```
[student@workstation ~]$ oc new-app --as-deployment-config \
> --name probes --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs \
> nodejs:12~https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps \
> --context-dir probes
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
...output omitted...
```

Observe que no hay espacio antes ni después del signo igual (=) después de `npm_config_registry`.

15. Vea los registros de compilación. Espere hasta que termine la compilación y la imagen del contenedor de aplicaciones se envíe al registro OpenShift:

```
[student@workstation ~]$ oc logs -f bc/probes
...output omitted...
STEP 8: RUN /usr/libexec/s2i/assemble
--> Installing application source ...
--> Building your Node application from source
...output omitted...
Push successful
```

- 1.6. Espere hasta que la aplicación se haya implementado. Consulte el estado del pod de la aplicación. El pod de la aplicación debe estar en el estado *Running* (En ejecución):

| NAME | READY | STATUS | RESTARTS | AGE |
|-----------------|------------|----------------|----------|-------|
| probes-1-build | 0/1 | Completed | 0 | 3m48s |
| probes-1-deploy | 0/1 | Completed | 0 | 2m21s |
| probes-1-vlzh2 | 1/1 | Running | 0 | 2m13s |

► 2. Pruebe manualmente el /ready de la aplicación y los extremos (endpoints) de /healthz.

- 2.1. Use una ruta para exponer la aplicación a acceso externo:

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

- 2.2. Pruebe el extremo (endpoint) /ready de la aplicación:

```
[student@workstation ~]$ curl \
> -i probes-${RHT_OCP4_DEV_USER}-probes.${RHT_OCP4_WILDCARD_DOMAIN}/ready
```

El extremo (endpoint) /ready simula un inicio lento de la aplicación, por lo que, durante los primeros 30 segundos después del inicio de la aplicación, devuelve un código de estado HTTP de 503 y la siguiente respuesta:

```
HTTP/1.1 503 Service Unavailable
...output omitted...
Error! Service not ready for requests...
```

Después de que la aplicación haya estado ejecutándose durante 30 segundos, devuelve lo siguiente:

```
HTTP/1.1 200 OK
...output omitted...
Ready for service requests...
```

2.3. Pruebe el extremo (endpoint) healthz de la aplicación:

```
[student@workstation ~]$ curl \
> -i probes-${RHT_OCP4_DEV_USER}-probes.${RHT_OCP4_WILDCARD_DOMAIN}/healthz
HTTP/1.1 200 OK
...output omitted...
OK
```

2.4. Pruebe la respuesta de la aplicación:

```
[student@workstation ~]$ curl \
> probes-${RHT_OCP4_DEV_USER}-probes.${RHT_OCP4_WILDCARD_DOMAIN}
Hello! This is the index page for the app.
```

► 3. Active los sondeos de disponibilidad y ejecución para la aplicación.

3.1. Use el comando oc set para agregar sondeos de ejecución y disponibilidad a la configuración de implementación.

```
[student@workstation ~]$ oc set probe dc/probes --liveness \
> --get-url=http://:8080/healthz \
> --initial-delay-seconds=2 --timeout-seconds=2
deploymentconfig.apps.openshift.io/probes probes updated
[student@workstation ~]$ oc set probe dc/probes --readiness \
> --get-url=http://:8080/ready \
> --initial-delay-seconds=2 --timeout-seconds=2
deploymentconfig.apps.openshift.io/probes probes updated
```

3.2. Compruebe el valor en las entradas livenessProbe y readinessProbe:

```
[student@workstation D0288-apps]$ oc describe dc/probes
Name: probes
...output omitted...
Liveness: http-get http://:8080/healthz delay=2s timeout=2s...
Readiness: http-get http://:8080/ready delay=2s timeout=2s...
...output omitted...
```

- 3.3. Espere a que el pod de aplicación vuelva a implementarse y tenga el estado Running (En ejecución):

```
[student@workstation D0288-apps]$ oc get pods
NAME READY STATUS RESTARTS AGE
...output omitted...
probes-3-hppqxi 0/1 Running 0 6s
```

El estado READY (LISTO) mostrará 0/1 si el valor AGE (VIGENCIA) es inferior a aproximadamente 30 segundos. Después de eso, el estado READY (LISTO) es 1/1:

```
[student@workstation D0288-apps]$ oc get pods
NAME READY STATUS RESTARTS AGE
...output omitted...
probes-3-hppqxi 1/1 Running 0 62s
```

- 3.4. Use el comando `oc logs` para ver los resultados de las sondas de ejecución y disponibilidad:

```
[student@workstation ~]$ oc logs -f dc/probes
...output omitted...
nodejs server running on http://0.0.0.0:8080
ping /healthz => pong [healthy]
ping /ready => pong [notready]
ping /healthz => pong [healthy]
ping /ready => pong [notready]
ping /healthz => pong [healthy]
ping /ready => pong [ready]
...output omitted...
```

Observe que el sondeo de disponibilidad falla durante aproximadamente 30 segundos después de la reimplementación y, luego, se realiza correctamente. Recuerde que la aplicación simula una inicialización lenta de la aplicación estableciendo, de manera forzada, un retraso de 30 segundos antes de responder con un estado Listo.

No finalice este comando. Continuará monitoreando la salida de este comando en el paso siguiente.

► 4. Simule una falla del sondeo de ejecución.

- 4.1. En una pestaña o ventana de terminal diferente, ejecute el script `~/D0288/labs/probes/kill.sh` para simular un error de sondeo de ejecución:

```
[student@workstation ~]$ ~/DO288/labs/probes/kill.sh
Switched app state to unhealthy...
```

- 4.2. Vuelva al terminal donde está monitoreando la implementación de la aplicación:

```
[student@workstation ~]$ oc logs -f dc/probes
...output omitted...
Received kill request. Changing app state to unhealthy...
ping /ready => pong [ready]
ping /healthz => pong [unhealthy]
ping /ready => pong [ready]
ping /healthz => pong [unhealthy]
ping /ready => pong [ready]
npm info lifecycle probes@1.0.0-poststart: probes@1.0.0
npm timing npm Completed in 150591ms
npm info ok
```

Observe que OpenShift reinicia el pod cuando el sondeo de ejecución falla algunas veces (el valor predeterminado es tres veces). Verá esta salida de registro solo cuando usted comprueba inmediatamente los registros de aplicaciones después de emitir la solicitud de finalización. Si verifica los registros después de que OpenShift reinicia el pod, los registros se borran y usted solo ve la salida que se exhibe en el siguiente paso.

- 4.3. Verifique si OpenShift reinicia el pod incorrecto. Continúe verificando la salida del comando `oc get pods`. Observe la columna RESTARTS (REINICIOS) y verifique que el conteo sea mayor que cero:

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
...output omitted...
probes-3-1tkkp   1/1     Running      1          21m
```

- 4.4. Vuelva a verificar los registros de aplicaciones para observar si el sondeo de ejecución se realizó correctamente y la aplicación informa un estado saludable:

```
[student@workstation ~]$ oc logs -f dc/probes
...output omitted...
ping /ready => pong [ready]
ping /healthz => pong [healthy]
...output omitted...
```

5. Verifique que el error del sondeo de ejecución se vea en el registro de eventos con el comando `oc describe` en el pod del paso anterior.

```
[student@workstation ~]$ oc describe pod/probes-3-1tkkp
Events:
  Type    Reason     Age   From      Message
  ----  -----  ----  ---      ---
...output omitted...
Warning  Unhealthy  ...   ...      Liveness probe failed: ... statuscode: 503
```

```
Normal  Killing    ...    ...    Container ... will be restarted
Normal  Created    ...    ...    Created container probes
Warning Unhealthy ...    ...    Readiness probe failed: ... statuscode: 503
```

Observe que OpenShift registra la falla del sondeo de ejecución y reinicia el pod después de que el sondeo falle tres veces. Debido a que la aplicación tiene un retraso de 30 segundos antes de que el sondeo de disponibilidad se realice correctamente, también verá una falla del sondeo de disponibilidad informada en el registro de eventos después de que OpenShift reinicia el pod.

- 6. Realice la limpieza. Elimine el proyecto `probes` en OpenShift:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-probes
```

Finalizar

En `workstation`, ejecute el comando `lab probes finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab probes finish
```

Esto concluye el ejercicio guiado.

Selección de la estrategia de implementación adecuada

Objetivos

Después de completar esta sección, deberá ser capaz de seleccionar la estrategia de implementación adecuada para una aplicación nativa de la nube.

Estrategias de implementación en OpenShift

Una *estrategia de implementación* es un método de cambiar o actualizar una aplicación. El objetivo es realizar cambios o actualizaciones con un tiempo de inactividad mínimo y un impacto reducido en los usuarios finales.

OpenShift ofrece diversas estrategias de implementación. Estas estrategias se pueden organizar en dos categorías principales. Un enfoque usa la configuración de implementación para la aplicación para definir la estrategia de implementación y el otro enfoque usa algunas funciones del enrutador OpenShift para enrutar tráfico a los pods de aplicaciones.

Las estrategias que cambian la configuración de implementación afectan todas las rutas que usan la aplicación. Las estrategias que usan las funciones de enrutador afectan a rutas individuales.

A continuación, se indican las estrategias que implican el cambio de la configuración de implementación:

Gradual (Rolling)

La estrategia gradual es la estrategia de implementación usada si no especifica una estrategia en la configuración de implementación para una aplicación.

Esta estrategia reemplaza de manera progresiva instancias de la versión anterior de una aplicación con instancias de la nueva versión. Esta estrategia ejecuta sondeos de disponibilidad para determinar cuándo están listos los nuevos pods antes de reducir (scaling down) los pods anteriores. Si tiene lugar un problema importante, se cancela la implementación gradual. La implementación también se puede cancelar en forma manual con el comando `oc rollout cancel`.

Las implementaciones graduales en OpenShift son *canary deployments* (implementaciones canario); se prueba una nueva versión (el canario) antes de reemplazar todas las instancias anteriores. Si el sondeo de disponibilidad no se puede realizar correctamente, se elimina la instancia canario y la configuración de implementación se revierte automáticamente.

Use una estrategia de implementación gradual cuando:

- No desea tener tiempo de inactividad durante la actualización de una aplicación.
- Su aplicación admite la ejecución de una versión anterior y una versión nueva al mismo tiempo.

Recrear (Recreate)

En esta estrategia, OpenShift primero detiene todos los pods que actualmente están en ejecución y recién entonces, inicia los pods con la nueva versión de la aplicación. Esta estrategia provoca tiempo de inactividad, ya que, durante un breve período, no se ejecuta ninguna instancia de su aplicación.

Use una estrategia de implementación de recrear cuando:

- Su aplicación no admite la ejecución de una versión anterior y una versión nueva al mismo tiempo.
- Su aplicación usa un volumen persistente con modo de acceso RWO (ReadWriteOnce), el cual no permite escrituras de varios pods.

Custom

Si ni las estrategias de implementación graduales ni las de recrear se adecuan a sus necesidades, puede usar la estrategia de implementación personalizada para implementar sus aplicaciones. Hay ocasiones en las que el comando que se va a ejecutar necesita un ajuste más preciso para el sistema (por ejemplo, memoria para la máquina virtual Java), o necesita usar una imagen personalizada con librerías desarrolladas internamente que no están disponibles para el público en general. Para estos tipos de casos de uso, use la estrategia personalizada. Puede proporcionar su propia imagen de contenedor personalizada, en la cual define el comportamiento de implementación. Esta imagen personalizada se define en el atributo `spec.strategy.customParams` de la configuración de implementación de la aplicación. También puede personalizar las variables de entorno y el comando que se ejecutará para la implementación.

Integración de implementaciones de OpenShift con hooks de ciclo de vida

Las estrategias Recreate (Recrear) y Rolling (Gradual) admiten *hooks de ciclo de vida*. Puede usar estos hooks para desencadenar eventos en puntos predefinidos en el proceso de implementación. Las implementaciones de OpenShift cuentan con tres hooks de ciclo de vida:

Hook previo al ciclo de vida

OpenShift ejecuta el hook previo al ciclo de vida antes de que se haya iniciado algún nuevo pod para una implementación, así como antes de que cualquier pod anterior se haya cerrado.

Hook en medio del ciclo de vida

El hook en medio del ciclo de vida se ejecuta después de que todos los pods antiguos en una implementación se hayan cerrado, pero antes de que se inicien nuevos pods. Los hooks en medio del ciclo de vida solo están disponibles para la estrategia Recreate (Recrear).

Hook posterior al ciclo de vida

El hook posterior al ciclo de vida se ejecuta después de que se hayan iniciado todos los nuevos pods para una implementación, así como después de que todos los pods anteriores se hayan cerrado.

Los hooks de ciclo de vida se ejecutan en contenedores separados, los cuales tienen una vida útil corta. OpenShift los limpia automáticamente después de que terminan de ejecutarse. La inicialización de la base de datos automática y las migraciones de bases de datos son buenos casos de uso para los hooks de ciclo de vida.

Cada hook cuenta con un atributo `failurePolicy`, que define la acción a tomar cuando se encuentra una falla en el hook. Existen tres políticas:

- Cancelar: El proceso de implementación se considera un error si falla el hook.
- Reintentar: Reintente la ejecución del hook hasta obtener un resultado satisfactorio.
- Ignorar: Ignore toda falla del hook y permita que continúe la implementación.

Implementación de estrategias avanzadas de implementación mediante el router OpenShift

A continuación, se enumeran las estrategias de implementación avanzada que usan las funciones de enrutador de OpenShift:

Implementaciones azul-verde (Blue-Green Deployment)

En las implementaciones azul-verde (Blue-Green), tiene dos entornos idénticos que se ejecutan al mismo tiempo, donde cada entorno ejecuta una versión diferente de la aplicación. El enrutador OpenShift se usa para dirigir el tráfico desde la versión actual en producción (verde) a la versión actualizada más reciente (azul). Puede implementar esta estrategia mediante una ruta y dos servicios. Defina un servicio para cada versión específica de la aplicación.

La ruta apunta a uno de los servicios en un momento determinado y se puede modificar para apuntar a un servicio diferente cuando esté listo, o para facilitar una reversión. Como desarrollador, puede probar la nueva versión de su aplicación conectándose al nuevo servicio antes de enrutar su tráfico de producción a este. Cuando su nueva versión de la aplicación está preparada para producción, cambie el enrutador de producción para que indique el nuevo servicio definido para su aplicación actualizada.

Implementación A/B (A/B Deployment)

La estrategia de implementación A/B le permite implementar una nueva versión de la aplicación para un conjunto limitado de usuarios en el entorno de producción. Puede configurar OpenShift para que enrute la mayoría de las solicitudes a la versión actualmente implementada en un entorno de producción, mientras que un número limitado de solicitudes se dirige a la nueva versión.

Al controlar la parte de las solicitudes enviadas a cada versión a medida que avanza la prueba, puede aumentar gradualmente el número de solicitudes enviadas a la nueva versión. Finalmente, puede detener el tráfico de enrutamiento a la versión anterior. A medida que ajusta la carga de solicitudes en cada versión, la cantidad de pods en cada servicio puede tener que escalarse para ofrecer el rendimiento esperado.

Las cosas adicionales a tener en cuenta al planear la estrategia de implementación de la aplicación incluyen la *compatibilidad con N-1* y la *finalización correcta*:

Compatibilidad con N-1

Muchas estrategias de implementación requieren la ejecución de dos versiones de la aplicación al mismo tiempo. Al ejecutar dos versiones de una aplicación al mismo tiempo, asegúrese de que la versión anterior del código pueda leer y gestionar (u omitir correctamente) los datos escritos por el nuevo código; esto se llama compatibilidad con N-1.

Los datos gestionados por la aplicación pueden adoptar muchas formas: datos almacenados en disco, en una base de datos o en un caché temporal. La mayoría de las aplicaciones web sin estado y bien diseñadas pueden admitir implementaciones graduales, pero es importante probar y diseñar la aplicación para admitir la compatibilidad N-1.

Finalización correcta

OpenShift permite que las instancias de aplicaciones se cierren correctamente antes de quitarlas de la lista de balanceo de carga del enrutador. Las aplicaciones deben asegurarse de finalizar de manera correcta las conexiones de usuarios antes de salir.

OpenShift envía una señal TERM a los procesos en el contenedor cuando desea cerrarlos. Al recibir una señal SIGTERM, el código de la aplicación debe dejar de aceptar nuevas conexiones. Detener nuevas conexiones garantiza que el enrutador pueda enrutar tráfico a otras instancias activas. A continuación, el código de aplicaciones debe esperar hasta que se hayan cerrado todas las conexiones (o se hayan finalizado correctamente las conexiones individuales en la próxima oportunidad) antes de salir.

Después de caducado el período de terminación correcta, OpenShift envía una señal KILL a cualquier proceso que no haya salido. Esto finaliza inmediatamente el proceso. El atributo `terminationGracePeriodSeconds` de un pod o una plantilla de pod controla el período de finalización correcto (el valor predeterminado es 30 segundos) y se puede personalizar por aplicación.



Referencias

Puede obtener más información sobre las estrategias de implementación en el capítulo *Implementaciones* de la guía *Aplicaciones de Red Hat OpenShift Container Platform 4.5*; en
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/applications/index#deployments

Una introducción a las implementaciones azul-verde, canario y gradual

<https://opensource.com/article/17/5/colorful-deployments>

Escenarios de implementación de lanzamiento de OpenShift

<https://keithtenzer.com/2016/08/11/openshift-v3-basic-release-deployment-scenarios/>

Estrategias de lanzamiento de aplicaciones con OpenShift

<https://access.redhat.com/articles/2897391>

Demostración de una estrategia de implementación personalizada con OpenShift

<https://www.youtube.com/watch?v=Plv17cvkEgQ>

Uso de tuberías en OpenShift para CI/CD

<https://developers.redhat.com/blog/2017/01/09/using-pipelines-in-openshift-3-3-for-cicd/>

Complemento (plug-in) de OpenShift Jenkins

<https://github.com/openshift/jenkins-plugin>

► Ejercicio Guiado

Aplicación de una estrategia de implementación

En este ejercicio, inicializará una base de datos mediante un hook de ciclo de vida de implementaciones.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Cambie la estrategia de implementación de una implementación de base de datos MySQL a Recreate (Recrear).
- Agregue un hook de ciclo de vida posterior a la implementación a la configuración de implementación para inicializar la base de datos MySQL con datos de un archivo SQL.
- Encuentre y corrija problemas de ejecución del ciclo de vida posterior a la implementación.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Imagen de contenedor MySQL 5.7 (rhscl/mysql-57-rhel7)

Ejecute el siguiente comando en `workstation` para validar los requisitos previos del ejercicio y descargar los archivos de trabajos de laboratorio y soluciones:

```
[student@workstation ~]$ lab strategy start
```

- 1. Cree un nuevo proyecto e implemente una aplicación, según la imagen de contenedor `rhscl/mysql-57-rhel7`, en el clúster OpenShift.

- 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 OpenShift con su nombre de usuario de desarrollador:

```
[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 para la aplicación. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-strategy
Now using project "youruser-strategy" on server "https://
api.cluster.domain.example.com:6443".
...output omitted...
```

- 1.4. Use el comando `oc new-app` para crear una nueva aplicación denominada `mysql`.

Copie o ejecute el comando del script `oc-new-app.sh` en la carpeta `~/D0288/labs/strategy`:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name mysql \
> -e MYSQL_USER=test -e MYSQL_PASSWORD=redhat -e MYSQL_DATABASE=testdb \
> --docker-image registry.access.redhat.com/rhscl/mysql-57-rhel7
--> Found Docker image ... "registry.access.redhat.com/rhscl/mysql-57-rhel7"
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
...output omitted...
```

- 1.5. Espere hasta que el pod de MySQL se haya implementado. Consulte el estado del pod. El pod se debe encontrar en estado *Running* (En ejecución):

```
[student@workstation ~]$ oc get pods
NAME        READY   STATUS    RESTARTS ...
mysql-1-deploy  0/1     Completed  0      ...
mysql-1-cx5hq   1/1     Running   0      ...
```

- 2. Aplique un parche a la configuración de implementación para cambiar la estrategia de implementación predeterminada a **Recreate** (Recrear).

- 2.1. Verifique que la estrategia de implementación predeterminada para la aplicación MySQL sea **Rolling** (Gradual):

```
[student@workstation ~]$ oc get dc/mysql -o jsonpath='{.spec.strategy.type}{"\n"}'
Rolling
```

- 2.2. En los siguientes pasos, realizará varios cambios a la configuración de implementación. Para evitar la reimplementación después de cada cambio, desactive los desencadenadores de cambio de configuración para la implementación:

```
[student@workstation ~]$ oc set triggers dc/mysql --from-config --remove
deploymentconfig.apps.openshift.io/mysql triggers updated
```

- 2.3. Cambie la estrategia de implementación predeterminada. Puede copiar el comando del script `~/D0288/labs/strategy/recreate.sh`, ejecutarlo o ingresar el comando, de la siguiente manera:

```
[student@workstation ~]$ oc patch dc/mysql --patch \
> '{"spec":{"strategy":{"type":"Recreate"}}}'
deploymentconfig.apps.openshift.io/mysql patched
```

- 2.4. Elimine el atributo `rollingParams` de la configuración de implementación. Puede copiar o ejecutar el comando del script `~/D0288/labs/strategy/rm-rolling.sh`:

```
[student@workstation ~]$ oc patch dc/mysql --type=json \
> -p='[{"op":"remove", "path": "/spec/strategy/rollingParams"}]' \
deploymentconfig.apps.openshift.io/mysql patched
```

- 3. Agregue un hook de ciclo de vida posterior para inicializar datos para la base de datos MySQL.
- 3.1. Revise brevemente el archivo `~/D0288/labs/strategy/users.sql` que contiene los datos para insertar en la base de datos MySQL. Este script de SQL crea una tabla denominada `users` (usuarios) e inserta tres filas de datos:

```
CREATE TABLE IF NOT EXISTS users (
    user_id int(10) unsigned NOT NULL AUTO_INCREMENT,
    name varchar(100) NOT NULL,
    email varchar(100) NOT NULL,
    PRIMARY KEY (user_id)) ENGINE=InnoDB DEFAULT CHARSET=utf8;

insert into users(name,email) values ('user1', 'user1@example.com');
insert into users(name,email) values ('user2', 'user2@example.com');
insert into users(name,email) values ('user3', 'user3@example.com');
```

- 3.2. Revise brevemente el script `~/D0288/labs/strategy/import.sh`, el cual descarga y ejecuta un script SQL para inicializar la base de datos. Este script se ejecuta en el hook de ciclo de vida posterior:

```
#!/bin/bash
...output omitted...

echo 'Downloading SQL script that initializes the database...'
curl -s -O https://github.com/RedHatTraining/D0288-apps/releases/download/
OCP-4.1-1/users.sql

echo "Trying $HOOK_RETRIES times, sleeping $HOOK_SLEEP sec between tries:"
while [ "$HOOK_RETRIES" != 0 ]; do

    echo -n 'Checking if MySQL is up...'
    if mysqlshow -h$MYSQL_SERVICE_HOST -u$MYSQL_USER -p$MYSQL_PASSWORD -P3306
$MYSQL_DATABASE &>/dev/null
    then
        echo 'Database is up'
        break
    else
        echo 'Database is down'

        # Sleep to wait for the MySQL pod to be ready
        sleep $HOOK_SLEEP
    fi

    let HOOK_RETRIES=$HOOK_RETRIES-1
done
```

```

done

if [ "$HOOK_RETRIES" = 0 ]; then
    echo 'Too many tries, giving up'
    exit 1
fi

# Run the SQL script
if mysql -h$MYSQL_SERVICE_HOST -u$MYSQL_USER -p$MYSQL_PASSWORD -P3306
$MYSQL_DATABASE < /tmp/users.sql
then
    echo 'Database initialized successfully'
else
    echo 'Failed to initialize database'
    exit 2
fi

```

El script intenta conectarse a la base de datos como máximo HOOK_RETRIES veces y permanece inactivo HOOK_SLEEP segundos entre intentos. El script debe reintentar y permanecer inactivo, ya que el pod para el hook se crea al mismo tiempo en que el pod del servidor de la base de datos está creando su base de datos y usuario.

- 3.3. Revise brevemente el script ~/D0288/labs/strategy/post-hook.sh, el cual agrega un nuevo hook de ciclo de vida posterior a la configuración de implementación:

```
[student@workstation ~]$ cat ~/D0288/labs/strategy/post-hook.sh
...output omitted...
oc patch dc/mysql --patch \
'{"spec": {"strategy": {"recreateParams": {"post": {"failurePolicy": "Abort", "execNewPod": {"containerName": "mysql", "command": ["/bin/sh", "-c", "curl -L -s https://github.com/RedHatTraining/D0288-apps/releases/download/OCP-4.1-1/import.sh -o /tmp/import.sh&&chmod 755 /tmp/import.sh&&/tmp/import.sh"]}}}}}'
```

Se ofrecen copias locales del import.sh y users.sql para su referencia. Estos archivos se descargan del servidor de contenido del aula durante la reimplementación, ya que el hook de ciclo de vida posterior se ejecuta en un pod separado.

- 3.4. Ejecute el script ~/D0288/labs/strategy/post-hook.sh:

```
[student@workstation ~]$ ~/D0288/labs/strategy/post-hook.sh
deploymentconfig.apps.openshift.io/mysql patched
```

- 4. Verifique la configuración de implementación con parche y despliegue la nueva configuración de implementación.
- 4.1. Verifique que la estrategia de implementación ahora sea Recreate (Recrear) y que exista un hook de ciclo de vida posterior que ejecute el script import.sh:

```
[student@workstation ~]$ oc describe dc/mysql | grep -A 3 'Strategy:'
Strategy: Recreate
Post-deployment hook (pod type, failure policy: Abort):
  Container: mysql
  Command: /bin/sh -c curl -L -s ...
```

- 4.2. Fuerce una nueva implementación para probar los cambios a la estrategia y el nuevo hook de ciclo de vida posterior:

```
[student@workstation ~]$ oc rollout latest dc/mysql
deploymentconfig.apps.openshift.io/mysql rolled out
```

- 4.3. Verifique que aparezca un nuevo pod MySQL con estado *Running* (En ejecución), seguido de la ejecución del segundo pod de implementación, así como el hook de ciclo de vida posterior en un pod separado:

```
[student@workstation ~]$ watch -n 2 oc get pods
NAME          READY   STATUS    RESTARTS   ...
mysql-2-deploy 1/1     Running   0          ...
mysql-2-hook-post 1/1     Running   0          ...
mysql-2-kbnpr  1/1     Running   0          ...
```

- 4.4. Después de unos segundos, el pod de hook de ciclo de vida posterior y el segundo pod de implementación fallan:

```
[student@workstation ~]$ watch -n 2 oc get pods
NAME          READY   STATUS    RESTARTS   ...
mysql-1-rcw95  1/1     Running   0          ...
mysql-2-deploy 0/1     Error     0          ...
mysql-2-hook-post 0/1     Error     0          ...
```

Cuando los pods de hook e implementación se encuentran en estado de error, presione **Ctrl+C** para salir del comando `watch`.

Observe que el pod `mysql-1-rcw95` es un pod nuevo. La segunda implementación finalizó el pod original de la primera implementación. Después de haber fallado la segunda implementación, se creó un nuevo pod mediante la configuración de implementación original de la primera implementación.

► 5. Encuentre y corrija problemas en el hook posterior al ciclo de vida.

- 5.1. Consulte los registros del pod fallido. Los registros muestran que el script intenta conectarse a la base de datos una única vez y, luego, informa un estado de error:

```
[student@workstation ~]$ oc logs mysql-2-hook-post
Downloading SQL script that initializes the database...
Trying 0 times, sleeping 2 sec between tries:
Too many tries, giving up
```

- 5.2. Observe que el script tiene un valor incorrecto de 0 para la variable `HOOK_RETRIES`, lo que hace que nunca se conecte a la base de datos. Aumente la cantidad de veces

que el script intenta conectarse al servidor de la base de datos. Establezca la variable de entorno HOOK_RETRIES en la configuración de implementación en un valor de 5.

```
[student@workstation ~]$ oc set env dc/mysql HOOK_RETRIES=5
deploymentconfig.apps.openshift.io/mysql updated
```

- Inicie una tercera implementación para ejecutar el hook una segunda vez, usando los nuevos valores para las variables de entorno:

```
[student@workstation ~]$ oc rollout latest dc/mysql
deploymentconfig.apps.openshift.io/mysql rolled out
```

- Espere hasta que el nuevo pod de hook posterior al ciclo de vida esté en estado Completado:

```
[student@workstation ~]$ watch -n 2 oc get pods
NAME          READY   STATUS    RESTARTS   ...
mysql-1-deploy 0/1     Completed  0          5m26s
mysql-2-deploy 0/1     Error     0          3m30s
mysql-2-hook-post 0/1     Error     0          3m8s
mysql-3-29jwt  1/1     Running   0          57s
mysql-3-deploy  0/1     Completed  0          79s
mysql-3-hook-post 0/1     Completed  0          48s
```

- Abra un nuevo terminal para ver los registros del pod posterior al hook de ciclo de vida. En ellos se muestra que el script puede conectarse a la base de datos algunas veces y, luego, informa un estado correcto:

```
[student@workstation ~]$ oc logs -f mysql-3-hook-post
Downloading SQL script that initializes the database...
Trying 5 times, sleeping 2 sec between tries:
Checking if MySQL is up...Database is up
mysql: [Warning] Using a password on the command line interface can be insecure.
Database initialized successfully
```

La cantidad de intentos depende de su hardware y de los nodos que cada pod está programado para ejecutar con OpenShift.

Si el hook se conecta la primera vez, el pod finaliza cuando intenta ver sus registros. Puede pasar al siguiente paso.

- Después de algunos segundos, el nuevo pod de base de datos es el que se crea usando los cambios más recientes a la configuración de implementación:

```
NAME          READY   STATUS    RESTARTS   ...
mysql-1-deploy 0/1     Completed  0          7m46s
mysql-2-deploy 0/1     Error     0          5m50s
mysql-2-hook-post 0/1     Error     0          5m28s
mysql-3-29jwt  1/1     Running   0          3m17s
mysql-3-deploy  0/1     Completed  0          3m39s
mysql-3-hook-post 0/1     Completed  0          3m8s
```

Después de que el pod mysql-3-hook-post se ejecuta y sale, presione Ctrl+C para salir del comando watch.

Observe que OpenShift conserva los pods fallidos, creados durante el intento de implementación anterior, para que pueda implementar sus registros para solucionar problemas.

► 6. Verifique que el nuevo pod de la base de datos MySQL contenga los datos del archivo SQL.

- 6.1. Abra una sesión de shell en el pod del contenedor MySQL. Use el comando `oc get pods` para obtener el nombre del pod MySQL actual:

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
...output omitted...
mysql-3-3p4m1  1/1     Running   0          8m
[student@workstation ~]$ oc rsh mysql-3-3p4m1
sh-4.2$
```

- 6.2. Verifique que la tabla `users` (usuarios) se haya creado y completado con datos del archivo SQL:

```
sh-4.2$ mysql -u$MYSQL_USER -p$MYSQL_PASSWORD $MYSQL_DATABASE
...output omitted...
Server version: 5.7.16 MySQL Community Server (GPL)
...output omitted...
mysql> select * from users;
+-----+-----+
| user_id | name   | email           |
+-----+-----+
|      1  | user1  | user1@example.com |
|      2  | user2  | user2@example.com |
|      3  | user3  | user3@example.com |
+-----+-----+
3 rows in set (0.00 sec)
```

- 6.3. Salga de la sesión MySQL y la shell del contenedor:

```
mysql> exit
Bye
sh-4.2$ exit
exit
[student@workstation ~]$
```

► 7. Realice la limpieza. Elimine el proyecto `strategy` en OpenShift:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-strategy
```

Finalizar

En `workstation`, ejecute el comando `lab strategy finish` para terminar este ejercicio. Esto es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab strategy finish
```

Esto concluye el ejercicio guiado.

Administración de implementaciones de aplicaciones con Comandos CLI

Objetivos

Después de completar esta sección, deberá ser capaz de administrar la implementación de una aplicación con los comandos de la CLI.

Configuración de implementación

Una configuración de implementación define la plantilla para un pod y administra la implementación de nuevas imágenes o de modificaciones en la configuración siempre que se modifiquen los atributos. Las configuraciones de implementación pueden admitir muchos patrones de implementación diferentes, incluidos un reinicio completo, actualizaciones continuas personalizables, así como hooks anteriores y posteriores al ciclo de vida.

OpenShift crea automáticamente un controlador de replicación que representa la plantilla del pod de configuración de implementación al crear una configuración de implementación. Cuando cambia la configuración de implementación, OpenShift crea un nuevo controlador de replicación con la plantilla de pod más reciente y se ejecuta un proceso de implementación para reducir verticalmente el controlador de replicación anterior y escalar verticalmente el nuevo controlador de replicación. OpenShift también agrega y quita automáticamente instancias de la aplicación tanto de los balanceadores de carga del servicio como de los enruteadores a medida que se inician o detienen.

Se declara una configuración de implementación dentro del atributo `DeploymentConfig` en un archivo de recursos, que puede ser en formato YAML o JSON. Use el comando `oc` para gestionar la configuración de implementación, como cualquier otro recurso OpenShift. En la siguiente plantilla, se muestra una configuración de implementación en formato YAML:

```
kind: "DeploymentConfig"
apiVersion: "v1"
metadata:
  name: "frontend" ①
spec:
  ...
  replicas: 5 ②
  selector:
    name: "frontend"
  triggers:
    - type: "ConfigChange" ③
    - type: "ImageChange" ④
      imageChangeParams:
        ...
  strategy:
    type: "Rolling"
  ...

```

① El nombre de la configuración de implementación.

② La cantidad de réplicas para ejecutar.

- ③ Un desencadenador de cambio de configuración que genera la creación de un nuevo controlador de replicación siempre que cambie la configuración de implementación.
- ④ Un desencadenador de cambio de imagen que genera la creación de un nuevo controlador de replicación cada vez que se dispone de una nueva versión de la imagen.

Gestión de implementaciones con comandos de la CLI

Se dispone de diversas opciones de línea de comando para gestionar implementaciones. En la siguiente lista, se describen las opciones disponibles:

- Para iniciar una implementación, use el comando `oc rollout latest dc/name`

```
[user@host ~]$ oc rollout latest dc/name
```

A menudo, se usa esta opción para iniciar una nueva implementación o actualizar una aplicación a la versión más reciente.

- Para ver el historial de implementaciones para una configuración de implementación específica, use el comando `oc rollout history dc/name`

```
[user@host ~]$ oc rollout history dc/name
```

- Para acceder a los detalles acerca de una implementación específica, anexe el parámetro `--revision` al comando `oc rollout history`:

```
[user@host ~]$ oc rollout history dc/name --revision=1
```

- Para acceder a los detalles acerca de una configuración de implementación y su revisión más reciente, use el comando `oc describe dc`:

```
[user@host ~]$ oc describe dc name
```

- Para cancelar una implementación, ejecute el comando `oc rollout cancel` con la opción `dc`:

```
[user@host ~]$ oc rollout cancel dc/name
```

Es posible que desee cancelar una implementación si tarda demasiado en iniciarse, hay incoherencias en el archivo de registro o la implementación afecta al comportamiento de otros recursos del sistema.



Advertencia

Cuando se cancele, la configuración de implementación se revierte automáticamente al controlador de replicación anterior en ejecución.

- Para volver a intentar una implementación que falló, ejecute el comando `oc rollout` con la opción `retry`:

```
[user@host ~]$ oc rollout retry dc/name
```

Puede volver a intentar una implementación después de cancelar previamente esa implementación o de encontrar un error que provoque un error en la implementación, si desea mantener la misma revisión.



nota

Cuando intenta nuevamente una implementación, se reiniciará el proceso de implementación pero no se creará una nueva revisión de la implementación. OpenShift también reinicia el controlador de replicación con la misma configuración que tenía cuando falló.

- Para usar una versión anterior de la aplicación, puede revertir la implementación con el comando `oc rollback`:

```
[user@host ~]$ oc rollback dc/name
```

Si hay un problema con la implementación más reciente, por ejemplo, que los usuarios se quejan de una nueva función que no funciona según lo esperado, puede revertir la implementación a una versión anterior que funcionaba de la aplicación mediante el comando `oc rollback`.



nota

Si no se especifica una revisión con el parámetro `--to-version`, se usa la última revisión implementada correctamente.



nota

Las configuraciones de implementación admiten la reversión automática a la última revisión exitosa de la configuración en caso de que falle el proceso de implementación más reciente. En este caso, la plantilla más reciente que no se pudo implementar permanece sin modificaciones y está disponible para revisión.

- Para evitar iniciar accidentalmente un nuevo proceso de implementación una vez completada una reversión, los desencadenadores de cambio de imagen se deshabilitan como parte del proceso de reversión. Sin embargo, después de una reversión, usted puede volver a habilitar los desencadenadores de cambio de imagen con el comando `oc set triggers`:

```
[user@host ~]$ oc set triggers dc/name --auto
```

- Para ver los registros de una implementación, use el comando `oc logs`:

```
[user@host ~]$ oc logs -f dc/name
```

Si la última revisión se está ejecutando o falló, el comando `oc logs` devuelve los registros del proceso responsable de implementar sus pods. De ser exitosa, devuelve los registros de un pod de su aplicación.

También puede ver registros de antiguos procesos de implementación fallidos, siempre y cuando no hayan sido eliminados o suprimidos en forma manual:

```
[user@host ~]$ oc logs --version=1 dc/name
```

- Puede escalar la cantidad de pods en una implementación usando el comando `oc scale`:

```
[user@host ~]$ oc scale dc/name --replicas=3
```

Eventualmente, la cantidad de réplicas se propaga al estado deseado y actual configurado por la configuración de implementación.

Desencadenadores de implementación

Una configuración de implementación puede contener *triggers* (desencadenadores), los cuales impulsan la creación de nuevas implementaciones en respuesta a eventos, tanto dentro como fuera de OpenShift. Existen dos tipos de eventos que desencadenan una implementación:

- Cambio en la configuración
- Cambio en la imagen

Desencadenador de cambio de configuración

El desencadenador `ConfigChange` genera una nueva implementación siempre que se detectan cambios en la plantilla del controlador de replicación de la configuración de implementación. Puede confiar en que este desencadenador se active después de cambiar el tamaño de la réplica, cambiar la imagen que se usará para la aplicación u otros cambios realizados en la configuración de implementación.

A continuación, se muestra un ejemplo de un desencadenador `ConfigChange`:

```
triggers:
  - type: "ConfigChange"
```

Desencadenador de cambio de imagen

El desencadenador `ImageChange` genera una nueva implementación siempre que cambie el valor de la etiqueta de un flujo de imágenes. Esto es útil en un entorno donde las imágenes se actualizan independientemente del código de la aplicación por motivos de seguridad o actualizaciones de librería.

A continuación, se muestra un ejemplo de un desencadenador `ImageChange`:

```
triggers:
  - type: "ImageChange"
    imageChangeParams:
      automatic: true①
      containerNames:
        - "helloworld"
      from:
        kind: "ImageStreamTag"
        name: "origin-ruby-sample:latest"
```

- ① Si el atributo `automatic` se establece como falso, el desencadenador se deshabilita.

En el ejemplo anterior, cuando cambia el valor de la etiqueta `latest` (más reciente) del flujo de imágenes `origin-ruby-sample`, se crea una nueva implementación con el nuevo valor de la etiqueta para el contenedor.

Use el comando `oc set triggers` para establecer un desencadenador de implementación para una configuración de implementación. Por ejemplo, para establecer el desencadenador `ImageChange`, ejecute el siguiente comando:

```
[user@host ~]$ oc set triggers dc/name \
> --from-image=myproject/origin-ruby-sample:latest -c helloworld
```

Establecer límites de recursos de implementación

Una implementación se completa por un pod que consume recursos (memoria y CPU) en un nodo. De manera predeterminada, los pods consumen recursos de nodos ilimitados. No obstante, si un proyecto especifica límites de recursos predeterminados, los pods solo consumen recursos hasta esos límites.

También puede limitar el uso de recursos especificando límites de recursos como parte de la estrategia de implementación. Estos límites de recursos se aplican a los pods de aplicación creados por la implementación, pero no a los pods del implementador. Puede usar recursos de implementación con las estrategias de implementación Recreate (Recrear), Rolling (Gradual) o Custom (Personalizada).

En el siguiente ejemplo, los recursos requeridos para la implementación se declaran bajo el atributo `resources` de la configuración de implementación:

```
type: "Recreate"
resources:
  limits:
    cpu: "100m" ❶
    memory: "256Mi" ❷
```

- ❶ Recurso CPU en unidades de CPU. `100m` equivale a 0.1 unidades de CPU.
- ❷ Recursos de memoria en bytes. `256Mi` equivale a 268435456 bytes ($256 * 2 ^ 20$).



Referencias

Para obtener información adicional sobre las implementaciones, consulte el capítulo *Implementaciones* de la guía *Aplicaciones para Red Hat OpenShift Container Platform 4.5* en https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html-single/applications/index#deployments

► Ejercicio Guiado

Administración de las implementaciones de aplicaciones

En este ejercicio, gestionará la implementación de una aplicación que se ejecuta en un clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implemente la aplicación basada en Thorntail en un clúster OpenShift.
- Actualice la configuración de implementación para la aplicación en ejecución para incluir un sondeo de ejecución.
- Realice cambios en la fuente de la aplicación y vuelva a implementar la aplicación.
- Revierta la aplicación a la versión implementada anteriormente.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen del compilador OpenJDK S2I `redhat-openjdk-18/openjdk18-openshift`.
- La aplicación quip en el repositorio Git.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos del ejercicio y descargar los archivos de trabajos de laboratorio y soluciones:

```
[student@workstation ~]$ lab app-deploy start
```

► 1. Revise el código fuente de la aplicación.

- 1.1. Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 1.2. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b app-deploy
Switched to a new branch 'app-deploy'
[student@workstation D0288-apps]$ git push -u origin app-deploy
...output omitted...
* [new branch]      app-deploy -> app-deploy
Branch app-deploy set up to track remote branch app-deploy from origin.
[student@workstation D0288-apps]$ cd ~
```

- 1.3. Inspeccione el archivo Quip.java en el directorio ~/D0288-apps/quip/src/main/java/com/redhat/training/example.

La aplicación quip es una implementación de servicio Java JAX-RS REST muy básica, con dos extremos (endpoints):

```
...output omitted...
@Path("/")
public class Quip {

    @GET
    @Produces("text/plain")
    public Response index() throws Exception {
        String host = InetAddress.getLocalHost().getHostName();
        return Response.ok("Veni, vidi, vici...\n").build();①
    }

    @GET
    @Path("/ready")
    @Produces("text/plain")
    public Response ready() throws Exception {
        return Response.ok("OK\n").build();②
    }
...output omitted...
```

- ① Las solicitudes al extremo (endpoint) de la URL raíz ('/') devuelven una comilla simple.
- ② Las solicitudes al extremo (endpoint) ready devuelven un mensaje OK.

► 2. Cree una aplicación basada en el código fuente.

- 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. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 2.3. Cree un nuevo proyecto para la aplicación. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-app-deploy
```

- 2.4. Cree una aplicación con el comando `oc new app`. El comando también se proporciona en el archivo `/home/student/D0288/labs/app-deploy/oc-new-app.sh`. Puede ejecutar este script directamente:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name quip \
> --build-env MAVEN_MIRROR_URL=http://${RHT_OCP4_NEXUS_SERVER}/repository/java \
> -i redhat-openjdk18-openshift:1.5 \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#app-deploy \
> --context-dir quip
--> Found image c1bf724 (9 months old) in image stream "openshift/redhat-
openjdk18-...
--> Creating resources ...
imagestream.image.openshift.io "quip" created
buildconfig.build.openshift.io "quip" created
deploymentconfig.apps.openshift.io "quip" created
service "quip" created
--> Success
...output omitted...
```

- 2.5. Vea los registros de compilación de la aplicación. Llevará algo de tiempo compilar la imagen del contenedor de aplicaciones y enviarla al registro interno OpenShift:

```
[student@workstation ~]$ oc logs -f bc/quip
...output omitted...
[INFO] Repackaged .war: /tmp/src/target/quip.war
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
...output omitted...
Pushing image ...-registry.svc:5000/youruser-app-deploy/quip:latest ...
...output omitted...
Push successful
```

- 2.6. Espere hasta que la aplicación se haya implementado. El pod de la aplicación debe estar en el estado *Running* (En ejecución), y marcado como listo:

```
[student@workstation ~]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
quip-1-build  0/1     Completed  0          2m17s
quip-1-deploy 1/1     Completed  0          53s
quip-1-v8gq4   1/1     Running   0          45s
```

► 3. Pruebe la aplicación para verificar que responda solicitudes de clientes.

- 3.1. Revise los registros de la aplicación para ver si hubo errores durante el inicio:

```
[student@workstation ~]$ oc logs dc/quip
...output omitted...
... INFO [org.jboss.as.server] (main) WFLYSRV0010: Deployed "quip.war" (...)
... INFO [org.wildfly.swarm] (main) WFSWARM99999: Thorntail is Ready
```

En los registros, se muestra que la aplicación se inició sin errores.

- 3.2. Verifique que el servicio OpenShift para la aplicación tenga un extremo (endpoint) registrado para enrutar solicitudes entrantes:

```
[student@workstation ~]$ oc describe svc/quip
Name:           quip
Namespace:      youruser-app-deploy
Labels:         app=quip
Annotations:   openshift.io/generated-by: OpenShiftNewApp
Selector:       app=quip,deploymentconfig=quip
Type:          ClusterIP
IP:            172.30.37.127
Port:          8080-tcp  8080/TCP
TargetPort:    8080/TCP
Endpoints:     10.128.2.111:8080
...output omitted...
```

- 3.3. Exponga la aplicación a un acceso externo usando una ruta:

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

- 3.4. Obtenga la URL de enrutamiento con el comando `oc get route`:

```
[student@workstation ~]$ oc get route/quip \
> -o jsonpath='{.spec.host}{"\n"}'
quip-youruser-app-deploy.apps.cluster.domain.example.com
```

- 3.5. Pruebe la aplicación con la URL de ruta que obtuvo en el paso anterior:

```
[student@workstation ~]$ curl \
> http://quip-${RHT_OCP4_DEV_USER}-app-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
Veni, vidi, vici...
```

► 4. Active los sondeos de disponibilidad y ejecución para la aplicación.

- 4.1. Use el comando `oc set` para agregar sondeos de ejecución y disponibilidad a la configuración de implementación.

```
[student@workstation ~]$ oc set probe dc/quip --liveness \
> --get-url=http://:8080/ready \
> --initial-delay-seconds=30 --timeout-seconds=2
deploymentconfig.apps.openshift.io/quip probes updated
[student@workstation ~]$ oc set probe dc/quip --readiness \
> --get-url=http://:8080/ready \
> --initial-delay-seconds=30 --timeout-seconds=2
deploymentconfig.apps.openshift.io/quip probes updated
```

4.2. Compruebe el valor en las entradas livenessProbe y readinessProbe:

```
[student@workstation ~]$ oc describe dc/quip
Name: quip
...output omitted...
  Liveness: http-get http://:8080/ready delay=30s timeout=2s...
  Readiness: http-get http://:8080/ready delay=30s timeout=2s...
...output omitted...
```

4.3. Espere a que el pod de aplicación vuelva a implementarse y tenga el estado Running (En ejecución):

```
[student@workstation ~]$ oc get pods
NAME      READY   STATUS    RESTARTS   AGE
...output omitted...
quip-3-hppqxi  1/1     Running   0          6s
```

4.4. Use el comando oc describe para inspeccionar el pod en ejecución y asegurarse de que los sondeos estén activos:

```
[student@workstation ~]$ oc describe pod quip-3-hppqxi | grep http-get
  Liveness: http-get http://:8080/ready delay=30s timeout=2s...
  Readiness: http-get http://:8080/ready delay=30s timeout=2s...
```

Los sondeos de disponibilidad y ejecución para la aplicación están activos.

4.5. Pruebe la aplicación con la URL de ruta que obtuvo en el paso anterior:

```
[student@workstation ~]$ curl \
> http://quip-${RHT_OCP4_DEV_USER}-app-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
Veni, vidi, vici...
```

► 5. Realice algunos cambios en la fuente de la aplicación. Verifique que puede ver los cambios cuando prueba la aplicación.

5.1. Revise el script provisto en ~/D0288/labs/app-deploy/app-change.sh. El script cambia el código fuente para imprimir el mensaje en inglés. El script cambia el mensaje y, luego, confirma y envía el cambio al repositorio Git del aula. Revise brevemente el script:

```
[student@workstation ~$ cat ~/D0288/labs/app-deploy/app-change.sh
#!/bin/bash

echo "Changing quip to english..."
sed -i 's/Veni, vidi, vici/I came, I saw, I conquered/g' \
/home/student/D0288-apps/quip/src/main/java/com/redhat/training/example/Quip.java

echo "Committing the changes..."
cd /home/student/D0288-apps/quip
git commit -a -m "Changed quip lang to english"

echo "Pushing changes to classroom Git repository..."
git push
cd ~
```

5.2. Ejecute el script ~/D0288/labs/app-deploy/app-change.sh:

```
[student@workstation ~]$ ~/D0288/labs/app-deploy/app-change.sh
Changing quip to english...
Committing the changes...
[app-deploy afdf7c3] Changed quip lang to english
...output omitted...
To https://github.com/youruser/D0288-apps
 dfe07f7..0aa1ac1 app-deploy -> app-deploy
```

5.3. Inicie una nueva compilación de la aplicación y siga los registros de compilación:

```
[student@workstation ~]$ oc start-build quip -F
build.build.openshift.io/quip-2 started
...output omitted...
Push successful
```

5.4. Espere hasta que un nuevo pod de la aplicación se haya implementado. El pod debe tener un estado *Running* (En ejecución). El pod también se debe marcar como listo, de la siguiente manera:

| NAME | READY | STATUS | RESTARTS | AGE |
|---------------|-------|----------------|----------|-------|
| quip-1-build | 0/1 | Completed | 0 | 5m |
| quip-1-deploy | 0/1 | Completed | 0 | 5m |
| quip-2-build | 0/1 | Completed | 0 | 2m45s |
| quip-3-deploy | 0/1 | Completed | 0 | 2m15s |
| quip-4-deploy | 0/1 | Completed | 0 | 41s |
| quip-4-vbdr4 | 1/1 | Running | 0 | 32s |

5.5. Vuelva a probar la aplicación después del cambio y verifique que esté impreso el mensaje en inglés:

```
[student@workstation ~]$ curl \
> http://quip-${RHT_OCP4_DEV_USER}-app-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
I came, I saw, I conquered...
```

- 6. Vuelva a la implementación anterior. Verifique que pueda ver el mensaje anterior (el que vio antes de realizar el cambio).
- 6.1. Revierta a la versión anterior del desarrollo. Verá un mensaje de advertencia de que los desencadenadores de cambio de imagen están desactivados por el comando `oc rollback`:

```
[student@workstation ~]$ oc rollback dc/quip
deploymentconfig.apps.openshift.io/quip deployment #5 rolled back to quip-3
Warning: the following images triggers were disabled: quip:latest
You can re-enable them with: oc set triggers dc/quip --auto
```

- 6.2. Espere a que se implemente el nuevo pod de la aplicación. Debe tener un estado *Running* (En ejecución). El pod también se debe marcar como listo, de la siguiente manera:

```
[student@workstation ~]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
quip-1-build  0/1     Completed  0          6m
quip-1-deploy 0/1     Completed  0          6m
quip-3-build  0/1     Completed  0          4m
quip-3-deploy 0/1     Completed  0          4m
quip-4-deploy 0/1     Completed  0          2m
quip-5-deploy  0/1     Completed  0          45s
quip-5-z7lg5   1/1     Running   0          17s
```

- 6.3. Despues de haber implementado la versión anterior de la aplicación, vuelva a probarla y verifique que esté impreso el mensaje en latín:

```
[student@workstation ~]$ curl \
> http://quip-${RHT_OCP4_DEV_USER}-app-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
Veni, vidi, vici...
```

- 7. Realice la limpieza. Elimine el proyecto `youruser-app-deploy` de OpenShift:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-app-deploy
project.project.openshift.io "youruser-app-deploy" deleted
```

Finalizar

En `workstation`, ejecute el comando `lab app-deploy finish` para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab app-deploy finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Administración de las implementaciones de aplicaciones

Lista de verificación de rendimiento

En este trabajo de laboratorio, gestionará la implementación de una aplicación y la escalará en un clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implementar una aplicación basada en PHP en un clúster OpenShift.
- Escalar la aplicación para ejecutar en múltiples pods.
- Modificar la fuente de la aplicación, volver a implementar la aplicación y verificar que se vean reflejados los cambios.
- Revertir un cambio y verificar que se haya implementado la versión anterior de la aplicación.

Antes De Comenzar

Para realizar este trabajo de laboratorio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- El código fuente de la aplicación `php-scale` en el repositorio Git.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos:

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

Requisitos

La aplicación se escribe en el PHP. La aplicación imprime un mensaje de cadena simple con la versión de la aplicación, y el nombre y la dirección IP del pod en el que se está ejecutando. Debe implementar y probar la aplicación en un clúster OpenShift según las siguientes instrucciones:

- Use el flujo de imágenes `php:7.3` para implementar la aplicación.
- Asegúrese de que el nombre de la aplicación para OpenShift sea `scale` (escalar).
- Cree la aplicación en un proyecto denominado `youruser-manage-deploy`.
- Asegúrese de que la aplicación sea accesible en la URL:

`http://scale-youruser-manage-deploy.apps.cluster.domain.example.com`.

- Asegúrese de que la aplicación use el repositorio Git en:

<https://github.com/youruser/D0288-apps>.

- El directorio `php-scale` en el repositorio Git contiene el código fuente para la aplicación.

Pasos

1. Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps  
[student@workstation D0288-apps]$ git checkout master  
...output omitted...
```

2. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b manage-deploy  
Switched to a new branch 'manage-deploy'  
[student@workstation D0288-apps]$ git push -u origin manage-deploy  
...output omitted...  
* [new branch]      manage-deploy -> manage-deploy  
Branch manage-deploy set up to track remote branch manage-deploy from origin.
```

3. Inicie sesión en el clúster OpenShift con su nombre de usuario de desarrollo personal. Cree un nuevo proyecto denominado `youruser-manage-deploy` e implemente la aplicación a través del flujo de imágenes `php:7.3`. Asegúrese de hacer referencia a la bifurcación `manage-deploy` que creó en el paso anterior al implementar la aplicación. Exponga y pruebe la aplicación usando la URL de la ruta. Verifique que puede ver `version 1` y el nombre de pod en la salida.
4. Verifique que la estrategia `Rolling (Gradual)` sea la estrategia de implementación predeterminada.
5. Inicie sesión en la consola web de OpenShift con su nombre de usuario de desarrollo personal. Escale la aplicación a dos pods. Use el comando `curl` para volver a probar la aplicación y verifique que las solicitudes cuenten con balanceo de carga mediante el método de Round Robin en los dos pods.
6. Cambie el número de versión en la fuente a 2. No realice ningún otro cambio al código fuente. Confirme los cambios al repositorio Git.
7. Use la consola web para iniciar una nueva compilación de la aplicación. Verifique que OpenShift reduzca (scale down) los pods que ejecutan la versión anterior y escale (scale up) los dos pods nuevos con la versión más reciente de la aplicación. Vuelva a probar la aplicación con el comando `curl`. Verifique que puede ver `version 2` en la salida.
8. Vuelva a la implementación anterior. Use el comando `curl` para volver a probar la aplicación. Verifique que puede ver `version 1` en la salida.
9. Califique su trabajo.
Ejecute el siguiente comando en `workstation` para verificar que se hayan realizado todas las tareas:

```
[student@workstation D0288-apps]$ lab manage-deploy grade
```

10. Limpie y elimine el proyecto.

Finalizar

En `workstation`, ejecute el script `lab manage-deploy finish` para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

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

Esto concluye el trabajo de laboratorio de revisión.

► Solución

Administración de las implementaciones de aplicaciones

Lista de verificación de rendimiento

En este trabajo de laboratorio, gestionará la implementación de una aplicación y la escalará en un clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implementar una aplicación basada en PHP en un clúster OpenShift.
- Escalar la aplicación para ejecutar en múltiples pods.
- Modificar la fuente de la aplicación, volver a implementar la aplicación y verificar que se vean reflejados los cambios.
- Revertir un cambio y verificar que se haya implementado la versión anterior de la aplicación.

Antes De Comenzar

Para realizar este trabajo de laboratorio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- El código fuente de la aplicación `php-scale` en el repositorio Git.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos:

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

Requisitos

La aplicación se escribe en el PHP. La aplicación imprime un mensaje de cadena simple con la versión de la aplicación, y el nombre y la dirección IP del pod en el que se está ejecutando. Debe implementar y probar la aplicación en un clúster OpenShift según las siguientes instrucciones:

- Use el flujo de imágenes `php:7.3` para implementar la aplicación.
- Asegúrese de que el nombre de la aplicación para OpenShift sea `scale` (escalar).
- Cree la aplicación en un proyecto denominado `youruser-manage-deploy`.
- Asegúrese de que la aplicación sea accesible en la URL:

`http://scale-youruser-manage-deploy.apps.cluster.domain.example.com`.

- Asegúrese de que la aplicación use el repositorio Git en:

<https://github.com/youruser/D0288-apps>.

- El directorio `php-scale` en el repositorio Git contiene el código fuente para la aplicación.

Pasos

1. Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

2. Cree una nueva bifurcación donde pueda guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b manage-deploy
Switched to a new branch 'manage-deploy'
[student@workstation D0288-apps]$ git push -u origin manage-deploy
...output omitted...
* [new branch]      manage-deploy -> manage-deploy
Branch manage-deploy set up to track remote branch manage-deploy from origin.
```

3. Inicie sesión en el clúster OpenShift con su nombre de usuario de desarrollo personal. Cree un nuevo proyecto denominado `youruser-manage-deploy` e implemente la aplicación a través del flujo de imágenes `php:7.3`. Asegúrese de hacer referencia a la bifurcación `manage-deploy` que creó en el paso anterior al implementar la aplicación. Exponga y pruebe la aplicación usando la URL de la ruta. Verifique que puede ver `version 1` y el nombre de pod en la salida.

- 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 D0288-apps]$ source /usr/local/etc/ocp4.config
```

- 3.2. Inicie sesión en OpenShift y cree un proyecto nuevo. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador.

```
[student@workstation D0288-apps]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation D0288-apps]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-manage-deploy
Now using project "yourusername-manage-deploy" on server "https://
api.cluster.domain.example.com:6443".
```

- 3.3. Cree una nueva aplicación:

```
[student@workstation D0288-apps]$ oc new-app --as-deployment-config --name scale \
> php:7.3~https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#manage-deploy \
> --context-dir php-scale
--> Found image f10275b (3 weeks old) in image stream "openshift/php" under...
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "scale" created
buildconfig.build.openshift.io "scale" created
deploymentconfig.apps.openshift.io "scale" created
service "scale" created
--> Success
...output omitted...
```

- 3.4. Vea los registros de compilación. Espere hasta que termine la compilación y la imagen del contenedor de aplicaciones se envíe al registro OpenShift:

```
[student@workstation D0288-apps]$ oc logs -f bc/scale
...output omitted...
Push successful
```

- 3.5. Espere hasta que la aplicación se haya implementado. Consulte el estado del pod de la aplicación. El pod de la aplicación debe estar en el estado *Running* (En ejecución):

```
[student@workstation D0288-apps]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
scale-1-build  0/1     Completed  0          5m14s
scale-1-deploy  0/1     Completed  0          82s
scale-1-w48nd  1/1     Running   0          74s
```

- 3.6. Use una ruta para exponer la aplicación a acceso externo:

```
[student@workstation D0288-apps]$ oc expose svc scale
route.route.openshift.io/scale exposed
```

- 3.7. Obtenga la URL de enrutamiento con el comando `oc get route`:

```
[student@workstation D0288-apps]$ oc get route scale \
> -o jsonpath='{.spec.host}{"\n"}'
scale-youruser-manage-deploy.apps.cluster.domain.example.com
```

- 3.8. Pruebe la aplicación con el comando `curl`:

```
[student@workstation D0288-apps]$ curl \
> http://scale-${RHT_OCP4_DEV_USER}-manage-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
This is version 1 of the app. I am running on host -> scale-1-gp3w0 (10.128.1.21)
```

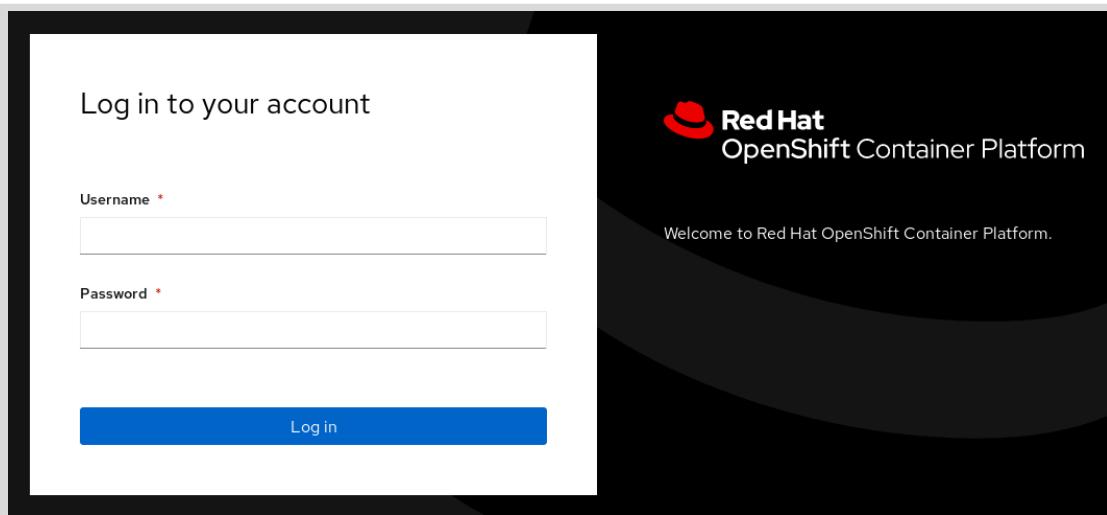
4. Verifique que la estrategia *Rolling* (Gradual) sea la estrategia de implementación predeterminada.

```
[student@workstation DO288-apps]$ oc describe dc/scale | grep 'Strategy'
Strategy: Rolling
```

5. Inicie sesión en la consola web de OpenShift con su nombre de usuario de desarrollo personal. Escale la aplicación a dos pods. Use el comando curl para volver a probar la aplicación y verifique que las solicitudes cuenten con balanceo de carga mediante el método de Round Robin en los dos pods.
 - 5.1. Abra un navegador web y navegue a la consola web de OpenShift. Inspeccione las rutas en el proyecto openshift-console para encontrar el nombre de host de su consola web de OpenShift.

```
[student@workstation DO288-apps]$ oc get route console -n openshift-console \
> -o jsonpath='{.spec.host}{"\n"}'
console-openshift-console.apps.cluster.domain.example.com
```

- 5.2. Inicie sesión con el usuario desarrollador. Su nombre de usuario (*youruser*) es la variable RHT_OCP4_DEV_USER en el archivo de configuración del aula /usr/local/etc/ocp4.config. Su contraseña es el valor de la variable RHT_OCP4_DEV_PASSWORD en el mismo archivo.



- 5.3. Haga clic en *youruser-manage-deploy* en la página **Projects** (Proyectos) para abrir la página **Overview** (Descripción general) para el proyecto. Haga clic en **Workloads** (Cargas de trabajo) para mostrar las implementaciones disponibles para el proyecto.
- 5.4. Seleccione la entrada de la configuración de implementación **scale** (escala) para mostrar sus detalles. En la sección **Details** (Detalles), haga clic en la flecha superior a la derecha del círculo azul para aumentar la cantidad de pods a dos.

The screenshot shows the OpenShift Project Details interface. The 'Workloads' tab is selected. A deployment configuration named 'scale' is listed, showing 1 pod. A 'Health Checks' alert is present, stating: 'Container scale does not have health checks to ensure your application is running correctly. Add Health Checks'. Below the deployment configuration, there are tabs for 'Details', 'Resources', and 'Monitoring'.

Observe cómo OpenShift crea otro pod. Esto podría demorar varios minutos.

- 5.5. Regrese a la ventana de terminal y use el comando `curl` para realizar varias solicitudes HTTP a la URL de ruta para probar la aplicación:

```
[student@workstation D0288-apps]$ curl \
> http://scale-${RHT_OCP4_DEV_USER}-manage-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
This is version 1 of the app. I am running on host -> scale-1-gp3w0 (10.128.1.21)
[student@workstation D0288-apps]$ curl \
> http://scale-${RHT_OCP4_DEV_USER}-manage-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
This is version 1 of the app. I am running on host -> scale-1-567x7 (10.129.1.101)
```

Debe ver las solicitudes con balanceo de carga mediante el método Round Robin en los dos pods.



nota

No puede usar un navegador web para probar la URL de ruta, porque el enrutador de OpenShift habilita las sesiones persistentes de forma predeterminada. Por lo tanto, no puede comprobar el comportamiento del balanceo de carga. Pruebe la aplicación con el comando `curl`.

6. Cambie el número de versión en la fuente a 2. No realice ningún otro cambio al código fuente. Confirme los cambios al repositorio Git.

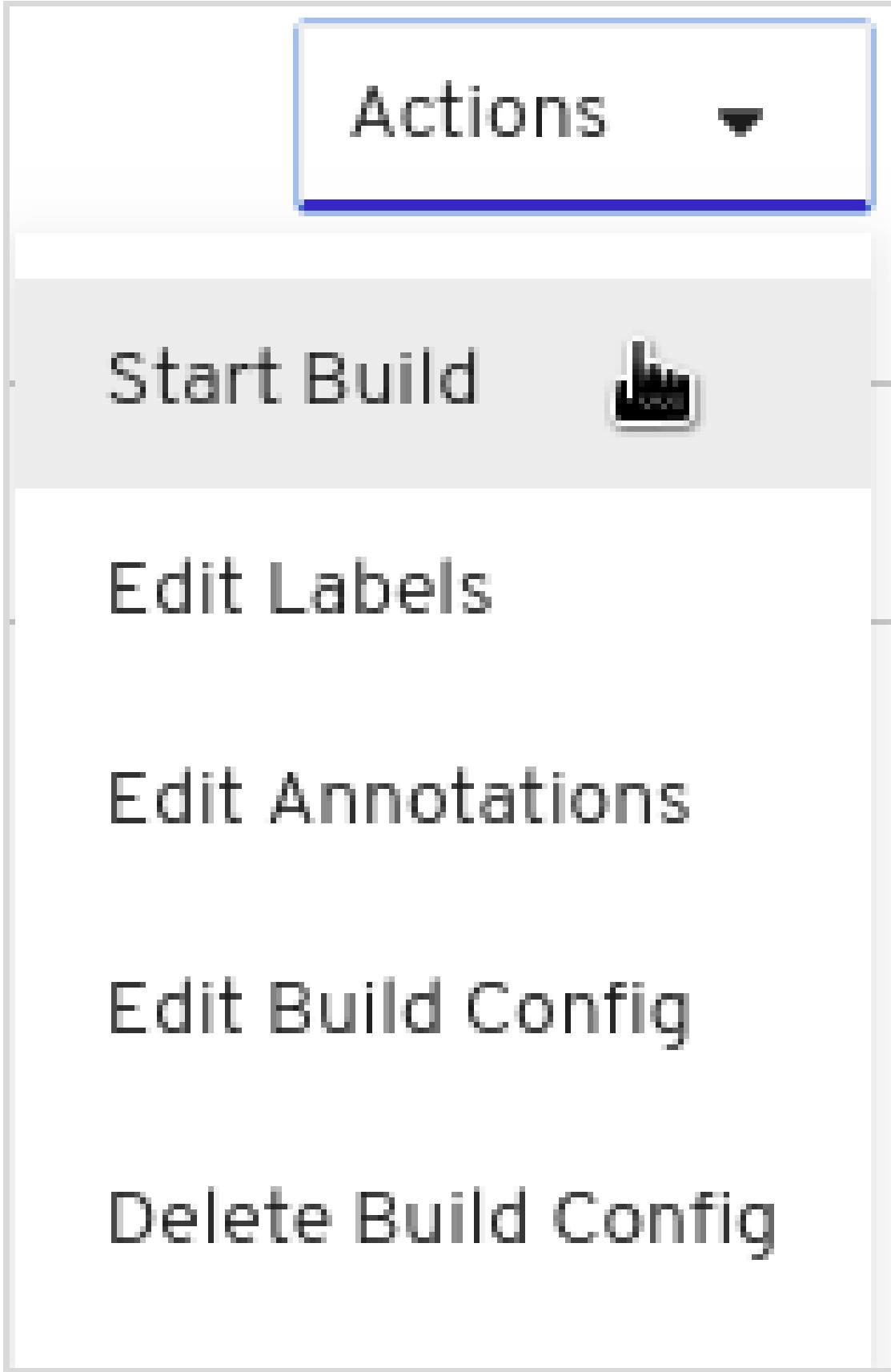
- 6.1. Edite el archivo `~/D0288-apps/php-scale/index.php` y cambie el número de versión (en la segunda línea) a 2, como se muestra a continuación. No realice ninguna otra modificación en este archivo:

```
<?php  
    print "This is version 2 of the app. I am running on host...  
?>
```

6.2. Confirme e inserte (push) los cambios en el repositorio Git:

```
[student@workstation D0288-apps]$ git commit -a -m "Updated app to version 2"  
[manage-deploy 3633f74] updating version  
 1 file changed, 1 insertion(+), 1 deletion(-)  
[student@workstation D0288-apps]$ git push
```

7. Use la consola web para iniciar una nueva compilación de la aplicación. Verifique que OpenShift reduzca (scale down) los pods que ejecutan la versión anterior y escala (scale up) los dos pods nuevos con la versión más reciente de la aplicación. Vuelva a probar la aplicación con el comando curl. Verifique que puede ver **version 2** en la salida.
 - 7.1. En el menú izquierdo de la consola web de OpenShift, abra el menú **Builds** (Compilaciones) y, a continuación, seleccione la entrada **Build Configs** (Configuración de compilación). Seleccione la configuración de compilación **scale** y, a continuación, use el menú desplegable **Actions** (Acciones) en la esquina superior derecha y haga clic en **Start Build** (Iniciar compilación):



- 7.2. La consola lo redirige a la página de resumen sobre la nueva compilación. Use el botón **Logs** (Registros) para ver el registro de compilación a medida que se crea la nueva imagen del contenedor de aplicaciones y se envía al registro interno de OpenShift.
- 7.3. Una vez terminada la compilación, OpenShift escala (scales up) dos nuevos pods de la nueva versión de la aplicación, verifica que los pods estén listos para recibir tráfico y, luego, reduce (scales down) los dos pods antiguos. Debe ver dos pods ejecutándose con la nueva versión.
- 7.4. Regrese a la ventana del terminal y use el comando `curl` para realizar varias solicitudes HTTP a la URL de ruta para probar la aplicación:

```
[student@workstation D0288-apps]$ curl \
> http://scale-${RHT_OCP4_DEV_USER}-manage-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
This is version 2 of the app. I am running on host -> scale-2-w7nfz (10.128.1.27)
[student@workstation D0288-apps]$ curl \
> http://scale-${RHT_OCP4_DEV_USER}-manage-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
This is version 2 of the app. I am running on host -> scale-2-dxswv (10.129.1.119)
```

Debe ver la versión 2 en la salida. También debe ver las solicitudes con balanceo de carga mediante el método Round Robin en los dos pods. Observe que el nombre del pod y las direcciones IP son diferentes de la implementación anterior.

8. Vuelva a la implementación anterior. Use el comando `curl` para volver a probar la aplicación. Verifique que puede ver `version 1` en la salida.
- 8.1. Revierta a la versión anterior del desarrollo. Recibirá un mensaje de advertencia de que los desencadenadores de cambio de imagen están desactivados por el comando `oc rollback`:

```
[student@workstation D0288-apps]$ oc rollback dc/scale
deploymentconfig.apps.openshift.io/scale deployment #3 rolled back to scale-1
Warning: the following images triggers were disabled: scale:latest
You can re-enable them with: oc set triggers dc/scale --auto
```

- 8.2. Espere a que se implemente el nuevo pod de la aplicación. Debe tener un estado *Running* (En ejecución). El pod también se debe marcar como listo, de la siguiente manera:

| NAME | READY | STATUS | RESTARTS | AGE |
|----------------|-------|----------------|----------|-------|
| scale-1-build | 0/1 | Completed | 0 | 40m |
| scale-1-deploy | 0/1 | Completed | 0 | 36m |
| scale-2-build | 0/1 | Completed | 0 | 10m |
| scale-2-deploy | 0/1 | Completed | 0 | 6m59s |
| scale-3-bcxpg | 1/1 | Running | 0 | 58s |
| scale-3-deploy | 0/1 | Completed | 0 | 77s |
| scale-3-lnp46 | 1/1 | Running | 0 | 68s |

- 8.3. Regrese a la ventana de terminal y use el comando `curl` para realizar varias solicitudes HTTP a la URL de ruta para probar la aplicación:

```
[student@workstation D0288-apps]$ curl \
> http://scale-${RHT_OCP4_DEV_USER}-manage-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
This is version 1 of the app. I am running on host -> scale-3-bcxpg (10.128.2.133)
[student@workstation D0288-apps]$ curl \
> http://scale-${RHT_OCP4_DEV_USER}-manage-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
This is version 1 of the app. I am running on host -> scale-3-lnp46 (10.131.1.206)
```

Debe ver la versión 1 en la salida. También debe ver las solicitudes con balanceo de carga mediante el método Round Robin en los dos pods. Observe que el nombre del pod y las direcciones IP son diferentes de la implementación anterior.

9. Califique su trabajo.

Ejecute el siguiente comando en **workstation** para verificar que se hayan realizado todas las tareas:

```
[student@workstation D0288-apps]$ lab manage-deploy grade
```

10. Limpie y elimine el proyecto.

```
[student@workstation D0288-apps]$ oc delete project \
> ${RHT_OCP4_DEV_USER}-manage-deploy
```

Finalizar

En **workstation**, ejecute el script **lab manage-deploy finish** para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

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

Esto concluye el trabajo de laboratorio de revisión.

Resumen

En este capítulo, aprendió lo siguiente:

- Los sondeos de disponibilidad y ejecución monitorean el estado de sus aplicaciones.
- Use la estrategia **Rolling** (Gradual) cuando pueda ejecutar dos versiones de la aplicación al mismo tiempo para actualizar la aplicación sin tiempo de inactividad. Esta estrategia primero escala (scale up) los pods adicionales con la nueva versión y, cuando está listo, reduce (scale down) los pods con la versión anterior.
- Use la estrategia **Recreate** (Recrear) cuando no pueda ejecutar dos versiones de su aplicación al mismo tiempo. Esta estrategia cierra todos los pods con la versión anterior y, luego, inicia pods adicionales con la versión más reciente.
- Use la estrategia **Custom** (Personalizada) para personalizar el proceso de implementación cuando las dos estrategias provistas por OpenShift no se adecuan a sus necesidades.
- Las estrategias **Recreate** (Recrear) y **Rolling** (Gradual) admiten hooks de ciclo de vida, el cual le permite personalizar la implementación en diversos puntos del proceso de implementación.
- Puede limitar el uso de recursos para implementaciones de aplicaciones especificando límites de recursos como parte de la estrategia de implementación.

capítulo 8

Implementación de tuberías de integración continua y de implementación continua en OpenShift

Meta

Crear e implementar tuberías de Jenkins para facilitar la integración y el despliegue continuos con OpenShift.

Objetivos

- Describir los principios de integración continua e implementación continua.
- Implementar Jenkins para administrar tuberías en OpenShift.
- Crear y ejecutar una tubería de Jenkins personalizada.

Secciones

- Descripción de conceptos de CI/CD (y cuestionario)
- Implementación de tuberías de Jenkins en OpenShift (y ejercicio guiado)
- Escritura de tuberías de Jenkins personalizadas (y ejercicio guiado)

Trabajo de laboratorio

Implementación de tuberías de integración continua y de implementación continua en OpenShift

Descripción de conceptos de CI/CD

Objetivos

Después de completar esta sección, debe poder describir los principios de integración continua e implementación continua.

Presentación de la integración continua, la entrega continua y el lanzamiento continuo

La *integración continua (CI)*, la *entrega continua (CD)* y el *lanzamiento continuo (CR)* son tres prácticas que se usan en los equipos de DevOps. La combinación de CI y CD, comúnmente denominada CI/CD, le permite entregar aplicaciones con frecuencia y confiabilidad a los clientes mediante la presentación de la automatización en las etapas de desarrollo e implementación. Estas dos técnicas son una solución al problema de la integración continua de nuevo código, que puede ser bastante engorroso y costoso cuando no se automatiza adecuadamente. La práctica de CR es un paso final opcional en el que se envían automáticamente los cambios a la producción después de una integración e implementación correctas de CI/CD.

CI/CD incluye la automatización continua y el monitoreo continua a lo largo del ciclo de vida de las aplicaciones, desde las fases de integración y pruebas hasta la entrega y la implementación. El proceso de implementación de esta automatización se puede denominar *tubería de CI/CD*. Idealmente, los equipos de desarrollo y operaciones trabajan juntos para compilar y respaldar esta automatización con metodologías ágiles.

Integración continua

La integración continua (CI) es un proceso de automatización dirigido a los desarrolladores para mejorar la calidad de su código y la fiabilidad de sus aplicaciones. CI funciona mejor cuando los desarrolladores integran su código en un repositorio de código fuente compartido como Git o Subversion varias veces al día. A continuación, cada confirmación se verifica mediante una compilación automatizada, lo que permite que los equipos detecten problemas a tiempo.

La automatización de las pruebas de aplicaciones permite que los desarrolladores eviten pasar un tiempo valioso validando manualmente cada cambio que realicen. CI también es una solución al problema de tener demasiadas bifurcaciones de una aplicación en desarrollo simultáneamente, donde las funciones de algunas de estas bifurcaciones pueden entrar en conflicto entre sí, o causar efectos secundarios no deseados.

En el modelo tradicional de cascada de desarrollo de software, los desarrolladores combinan el código fuente de sus bifurcaciones en la bifurcación principal con poca frecuencia y normalmente los integran todos a la vez (este enfoque se llama "integración de big bang"). El proceso resultante puede ser tedioso, manual, propenso a errores y requerir mucho tiempo, ya que la combinación de código a menudo tiene muchos conflictos.

Un mejor enfoque es adoptar un proceso más ágil de integración continua, donde los desarrolladores confirman regularmente los cambios para que se prueben y se verifiquen automáticamente. En un proyecto ágil típico, las tareas se dividen en unidades de trabajo mucho más pequeñas que se completan más fácilmente en un marco de tiempo corto, lo que permite que los miembros del equipo integren su trabajo con todo el proyecto hasta varias veces al día, lo que proporciona una serie de beneficios para el proyecto, como:

- Las tareas más pequeñas son más fáciles de reasignar y tienen resultados que generalmente son más fáciles de probar de forma automatizada.
- Si las pruebas automatizadas detectan un conflicto entre el código nuevo y el existente, la tubería de CI facilita la identificación y corrección rápida de esos errores.
- Las pequeñas tareas contribuyen a la transparencia del progreso del proyecto, lo que permite que todos se mantengan al día con el diseño y la dirección de la aplicación.
- Las tareas pequeñas que fallan suelen producir menos daño al proyecto general en forma de retrasos o efectos negativos para el código base al requerir una gran cantidad de refactorización.

Este enfoque tiene algunos requisitos claros para funcionar de forma fiable:

- Los desarrolladores deben aceptar usar un repositorio de administración de código fuente (SCM): su código fuente se combina regularmente en una base de código común en ese repositorio.
- Debe haber pruebas de unidades suficientemente extensas para cada componente individual desarrollado a fin de garantizar que un solo componente funcione de acuerdo con las especificaciones. Algunos equipos deciden usar el diseño controlado por pruebas (TDD) para cumplir con este requisito.
- Debe haber pruebas de integración muy básicas, pero representativas, que ejerzan las funciones lógicas de varios módulos en conjunto, como realizar una compra. Estas pruebas suelen realizarse en la capa más baja posible, como simplemente invocar componentes de lógica de negocios internamente en la aplicación y simular cualquier dependencia externa, como bases de datos y otros servicios web.

Una tubería de CI de ejemplo podría ser así:

1. La tubería se desencadena mediante una nueva confirmación en el repositorio de código fuente.
2. La tubería compila toda la aplicación directamente desde el código fuente. Las herramientas de Lint y análisis de código estático se ejecutan en esta etapa.
3. Las pruebas de unidad e integración se ejecutan para comprobar que el nuevo código no interrumpe la funcionalidad existente ni introduzca nuevos errores.
4. Si todas las etapas se completan correctamente, la aplicación estará lista para su implementación en un entorno de prueba o ensayo, donde se realizan más pruebas de extremo a extremo (tanto manuales como automatizadas) y, si se aprueba, el equipo puede promover automáticamente la aplicación a producción.

Entrega continua

El objetivo de la entrega continua (CD) es tener código en un estado que esté siempre listo para la implementación en un entorno de producción.

En el proceso de cascada tradicional, la implementación de código en producción es un proceso manual complejo, tedioso, con listas de comprobación largas, lo que implica el uso de varios equipos y mucha depuración después de implementar el código para garantizar que funcione según lo previsto.

En una tubería de entrega continua, cada etapa, desde la combinación de cambios de código hasta la entrega de compilaciones listas para producción, implica la automatización de pruebas

y la automatización de versiones de código. Al final del proceso, el equipo de operaciones puede implementar con confianza pequeños cambios incrementales en una aplicación en producción de forma rápida y eficiente.

Para que la entrega continua sea eficaz, es esencial que la tubería ya contenga un proceso de integración continua. Sin pruebas automatizadas ni verificación fiables de su aplicación, no puede implementarla continuamente con confianza. Una tubería de CD de ejemplo podría ser así:

1. Suponiendo que todas las etapas de la tubería de integración continua sean correctas, implemente la aplicación en un entorno de control de calidad o de ensayo.
2. Ejecute pruebas automatizadas y manuales de extremo a extremo en el entorno de ensayo. Puede agregar una etapa opcional en la que un operador humano comprueba manualmente que las pruebas se han completado y que la funcionalidad de la aplicación cumple las expectativas de los usuarios finales.
3. De manera opcional, implemente ("promueva") la aplicación en el entorno de producción.

Lanzamiento continuo

El objetivo del lanzamiento continuo (CR), también llamado "implementación continua" (que no debe confundirse con la entrega continua), es enviar todos los cambios a los usuarios finales de la forma más rápida y eficiente posible. Lograr CR significa reemplazar cuidadosamente los pasos de implementación manual, tras la aprobación de producción, por los automatizados, de modo que se mantengan los requisitos de nivel de servicio. En lugar de que alguien (o un equipo de personas) coordine cada paso de la implementación de aplicaciones en el entorno de producción, estos pasos deben implementarse de forma automatizada.

Hay una serie de estrategias adicionales que facilitan la publicación continua exitosa, incluidas, entre otras:

- Entornos de producción azul-verde (Blue-Green), donde las pruebas de humo posteriores a la implementación se realizan en el entorno inactivo antes de la sustitución.
- Escenarios de prueba A/B en las que se pueden ejecutar varias versiones diferentes de una aplicación al mismo tiempo, y solo una pequeña parte de los clientes está expuesta a la nueva versión durante un tiempo, lo que permite una reversión rápida y un daño mínimo en caso de problemas.
- Diferentes estrategias de implementación (como implementaciones continuas), donde los pasos exactos de un lanzamiento dependen de la naturaleza de la aplicación, e incluso posibles cambios en la estructura de datos, cualquiera de los cuales puede impedir que podamos ejecutar más de una versión (o incluso más de una instancia) de la aplicación al mismo tiempo.
- Aplicaciones compuestas donde diferentes módulos de back-end sirven a cada uno de los componentes de la interfaz de usuario (como el catálogo, la vista previa del ítem, las sugerencias, etc.), lo que permite realizar actualizaciones independientes de módulos individuales.
- Codificación de aplicaciones defensivas, incluido el uso de patrones de tolerancia a errores de microservicio, como el disyuntor, y las respuestas de reserva en caso de que falte una dependencia.
- Pruebas de humo claras y precisas, junto con procedimientos de reversión fiables para cada paso de la implementación de la aplicación.
- Monitoreo continua funcional y del rendimiento para cada módulo de aplicación y cada función lógica.

Es importante recordar que, mientras que CR es una práctica maravillosa si se implementa, puede no ser adecuada para todo. Hay aplicaciones, particularmente las monolíticas heredadas, pero también otras, que no se pueden modular fácilmente, no se pueden escalar fácilmente en sentido horizontal, por lo que constituyen una base deficiente para una implementación continua típica sin tiempo de inactividad.

Incluso hay casos de uso donde la implementación de CR trae más problemas de lo que vale. Considere un entorno de back-office sin conexión típico, donde la empresa opera en un cronograma muy predecible, dando al equipo de operaciones una gran oportunidad para interrupciones programadas, en lugar de arriesgarse a la implementación de una nueva versión durante las horas de trabajo, lo que podría perturbar a cientos o miles de clientes. Sin embargo, incluso en estos entornos, tener algunos (o la mayoría) de los bloques de compilación para CR en su lugar puede simplificar y acelerar enormemente el proceso de implementación manual, y hacerlo más confiable.



Referencias

¿Qué es CI/CD?

<https://www.redhat.com/en/topics/devops/what-is-ci-cd>

CI/CD con OpenShift

<https://blog.openshift.com/cicd-withOpenshift/>

► Cuestionario

Conceptos de CI/CD

Elija las respuestas correctas para las siguientes preguntas:

► 1. Organice las siguientes etapas en una tubería de integración continua para una aplicación Java en el orden correcto:

1. Compile el código con Apache Maven y, a continuación, ejecute una herramienta de análisis de código para asegurarse de que el código cumpla con los estándares de codificación en Java de sus organizaciones.
2. Etiquete el código fuente. Cree un archivo zip del repositorio de código y guárdelo como un artefacto en un servidor Nexus, para que pueda compilar repetidamente la aplicación en el entorno de ensayo.
3. Configure hooks web para habilitar el inicio de la tubería cuando el código se confirma en el repositorio de código fuente.
4. Ejecute pruebas de JUnit y pruebas de integración.
 - a. 3, 2, 4, 1
 - b. 2, 3, 1, 4
 - c. 3, 1, 4, 2
 - d. 3, 2, 1, 4

► 2. ¿Cuáles dos de las etapas siguientes son buenos candidatos para una tubería de integración continua que proporcione rápidamente microservicios de Node.js? (Elija dos opciones).

- a. Ejecutar las herramientas ESLint en toda la base de código e informar de los problemas.
- b. Ejecutar las herramientas de webpack en la base de código e implementar el microservicio en un servidor de red de entrega de contenido (CDN) que aloje archivos estáticos.
- c. Agregar una dependencia de administrador de paquetes de nodos (NPM) al proyecto que reinicie automáticamente el servidor Node.js cuando se detectan cambios en el código.
- d. Ejecutar `npm test` en el proyecto para ejecutar todas las pruebas de unidades.
- e. Ejecutar `npm db-generate` para generar un código de acceso de base de datos con scaffolding (soporte) para el proyecto.

► **3. Organice las siguientes etapas en el orden correcto para una tubería de integración continua para una aplicación Python:**

1. Ejecutar las herramientas de `pylint` en el proyecto Python e informar de los problemas.
2. Ejecutar las pruebas de unidad con el marco (framework) `pytest`.
3. Agregar etiquetas Git al proyecto para habilitar compilaciones reproducibles.
4. Comprar código fuente de aplicación desde un repositorio Git.
 - a. 4, 2, 3, 1
 - b. 4, 3, 1, 2
 - c. 4, 3, 2, 1
 - d. 4, 1, 2, 3

► **4. ¿Cuáles dos de las etapas siguientes son buenos candidatos para una tubería de integración continua que proporcione rápidamente microservicios de PHP implementados en un servidor web Apache? (Elija dos opciones).**

- a. Reiniciar automáticamente el servidor web Apache cuando se detecten cambios en el código fuente.
- b. Ejecutar las pruebas de unidad con el marco (framework) `PHPUnit`.
- c. Ajustar automáticamente el servidor web Apache en función del aumento o la disminución de la carga.
- d. Formatear y sangrar automáticamente el código para cumplir con los estándares de codificación de PHP 7.
- e. Enviar un correo electrónico cuando los usuarios finales se encuentran con un estado de error en la aplicación que da como resultado un estado HTTP 500 - Server Error (Error de servidor).
- f. Ejecutar trabajos cron a intervalos establecidos para recopilar estadísticas sobre los usuarios que acceden a la aplicación.

► Solución

Conceptos de CI/CD

Elija las respuestas correctas para las siguientes preguntas:

► 1. Organice las siguientes etapas en una tubería de integración continua para una aplicación Java en el orden correcto:

1. Compile el código con Apache Maven y, a continuación, ejecute una herramienta de análisis de código para asegurarse de que el código cumpla con los estándares de codificación en Java de sus organizaciones.
2. Etiquete el código fuente. Cree un archivo zip del repositorio de código y guárdelo como un artefacto en un servidor Nexus, para que pueda compilar repetidamente la aplicación en el entorno de ensayo.
3. Configure hooks web para habilitar el inicio de la tubería cuando el código se confirma en el repositorio de código fuente.
4. Ejecute pruebas de JUnit y pruebas de integración.
 - a. 3, 2, 4, 1
 - b. 2, 3, 1, 4
 - c. 3, 1, 4, 2
 - d. 3, 2, 1, 4

► 2. ¿Cuáles dos de las etapas siguientes son buenos candidatos para una tubería de integración continua que proporcione rápidamente microservicios de Node.js? (Elija dos opciones).

- a. Ejecutar las herramientas ESLint en toda la base de código e informar de los problemas.
- b. Ejecutar las herramientas de webpack en la base de código e implementar el microservicio en un servidor de red de entrega de contenido (CDN) que aloje archivos estáticos.
- c. Agregar una dependencia de administrador de paquetes de nodos (NPM) al proyecto que reinicie automáticamente el servidor Node.js cuando se detectan cambios en el código.
- d. Ejecutar `npm test` en el proyecto para ejecutar todas las pruebas de unidades.
- e. Ejecutar `npm db-generate` para generar un código de acceso de base de datos con scaffolding (soporte) para el proyecto.

► **3. Organice las siguientes etapas en el orden correcto para una tubería de integración continua para una aplicación Python:**

1. Ejecutar las herramientas de `pylint` en el proyecto Python e informar de los problemas.
2. Ejecutar las pruebas de unidad con el marco (framework) `pytest`.
3. Agregar etiquetas Git al proyecto para habilitar compilaciones reproducibles.
4. Comprar código fuente de aplicación desde un repositorio Git.
 - a. 4, 2, 3, 1
 - b. 4, 3, 1, 2
 - c. 4, 3, 2, 1
 - d. 4, 1, 2, 3

► **4. ¿Cuáles dos de las etapas siguientes son buenos candidatos para una tubería de integración continua que proporcione rápidamente microservicios de PHP implementados en un servidor web Apache? (Elija dos opciones).**

- a. Reiniciar automáticamente el servidor web Apache cuando se detecten cambios en el código fuente.
- b. Ejecutar las pruebas de unidad con el marco (framework) `PHPUnit`.
- c. Ajustar automáticamente el servidor web Apache en función del aumento o la disminución de la carga.
- d. Formatear y sangrar automáticamente el código para cumplir con los estándares de codificación de PHP 7.
- e. Enviar un correo electrónico cuando los usuarios finales se encuentran con un estado de error en la aplicación que da como resultado un estado HTTP 500 - Server Error (Error de servidor).
- f. Ejecutar trabajos cron a intervalos establecidos para recopilar estadísticas sobre los usuarios que acceden a la aplicación.

Implementación de tuberías de Jenkins en OpenShift

Objetivos

Después de completar esta sección, debería poder implementar Jenkins para administrar tuberías en OpenShift.

Presentación de Jenkins en OpenShift

OpenShift proporciona compatibilidad para crear, implementar y administrar tuberías de CI/CD mediante Jenkins. Puede crear tuberías complejas de CI/CD que automatizan el flujo de trabajo de la implementación rápida de la aplicación en producción.

OpenShift viene con compatibilidad integrada para ejecutar tuberías de Jenkins. Las tuberías describen el proceso automatizado para compilar, probar, implementar y promocionar sus aplicaciones en OpenShift. Las tuberías se describen mediante un *Jenkinsfile* (Archivo de Jenkins), que encapsula los pasos de automatización de la tubería.

Jenkins proporciona un complemento (plug-in) de tubería que ofrece una función para describir los pasos de tubería mediante un lenguaje específico de dominio (DSL) basado en el lenguaje de programación Groovy. OpenShift incluye varios otros complementos (plug-ins) para Jenkins que pueden ayudarle a crear, implementar y ejecutar tuberías para cualquier tipo de aplicación que desee implementar.

Lista de complementos (plug-ins) de Jenkins

Complemento (plug-in) de cliente de OpenShift Jenkins

Jenkins Client Plugin (Complemento de cliente de Jenkins) tiene como objetivo proporcionar un lenguaje específico de dominio simple y conciso, con la sintaxis de tubería de Jenkins para interacciones enriquecidas con OpenShift. El complemento (plug-in) aprovecha la herramienta de línea de comandos de OpenShift (oc), que debe estar disponible en los nodos esclavos de Jenkins que ejecutan la tubería. Los nodos esclavos de Jenkins son imágenes de contenedor que se pueden ejecutar en OpenShift.

Este complemento (plug-in) se instala y habilita de forma predeterminada cuando se usa la imagen de Jenkins integrada de OpenShift Container Platform. El complemento (plug-in) habilita las funciones específicas de OpenShift Container Platform, que están disponibles para su uso en el archivo de Jenkins de la tubería.

Complemento (plug-in) de sincronización de OpenShift Jenkins

El Complemento de sincronización de OpenShift Jenkins actúa como puente entre el clúster OpenShift y la instancia de Jenkins, y mantiene la configuración de compilación y los objetos de compilación sincronizados con los trabajos y las compilaciones de Jenkins.

Complemento (plug-in) de inicio de sesión de OpenShift

Login Plugin (Complemento de inicio de sesión) de OpenShift integra la autenticación y la autorización de la instancia de Jenkins con el clúster OpenShift, lo que proporciona una funcionalidad de inicio de sesión único. Debe acceder al panel de Jenkins con las mismas credenciales que usa para la consola web de OpenShift.

Ejecución de tuberías de Jenkins en OpenShift

OpenShift proporciona *JenkinsPipelineStrategy* como un tipo de compilación que le permite definir una tubería para que la ejecute Jenkins. OpenShift puede iniciar, monitorear y administrar la compilación de la misma manera que cualquier otro tipo de compilación.

Los flujos de trabajo de tubería se definen en un archivo de Jenkins, ya sea incrustado directamente en la configuración de compilación o proporcionado en un repositorio Git, al que se hace referencia en la configuración de compilación. Se recomienda mantener el archivo de Jenkins como un archivo independiente en la raíz del árbol de código fuente de la aplicación y hacer referencia a él en la configuración de compilación.



nota

La estrategia de compilación de la tubería de Jenkins quedó obsoleta en OpenShift Container Platform 4. La funcionalidad equivalente está presente en las tuberías de OpenShift basadas en Tekton. A pesar de la obsolescencia, las imágenes de Jenkins en OpenShift siguen siendo totalmente compatibles.

Con los siguientes pasos se creará y se ejecutará una tubería simple de Jenkins en OpenShift:

1. Primero debe implementar una instancia de Jenkins en el clúster OpenShift. Cree un proyecto para alojar la instancia de Jenkins:

```
[user@host ~]$ oc new-project myjenkins
```

2. OpenShift viene con plantillas listas para usar para implementar Jenkins:

```
[user@host ~]$ oc get templates -n openshift | grep jenkins
jenkins-ephemeral      Jenkins service, without persistent storage....
jenkins-persistent      Jenkins service, with persistent storage....
```

3. Implemente la instancia de Jenkins:

```
[user@host ~]$ oc new-app --as-deployment-config jenkins-ephemeral
```

Jenkins tardará un tiempo en implementarse. Espere hasta que los pods de Jenkins estén en ejecución y listos.

```
[user@host ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
jenkins-1-deploy  0/1     Completed  0          13d
jenkins-1-rtnw5  1/1     Running   0          13d
```

4. Obtenga la dirección URL de ruta para el panel de Jenkins:

```
[user@host ~]$ oc get route
jenkins-myjenkins.apps.cluster.domain.example.com
```

5. Vaya a la dirección URL de ruta del paso anterior desde un navegador web. Se le solicitará que inicie sesión con sus credenciales de OpenShift.

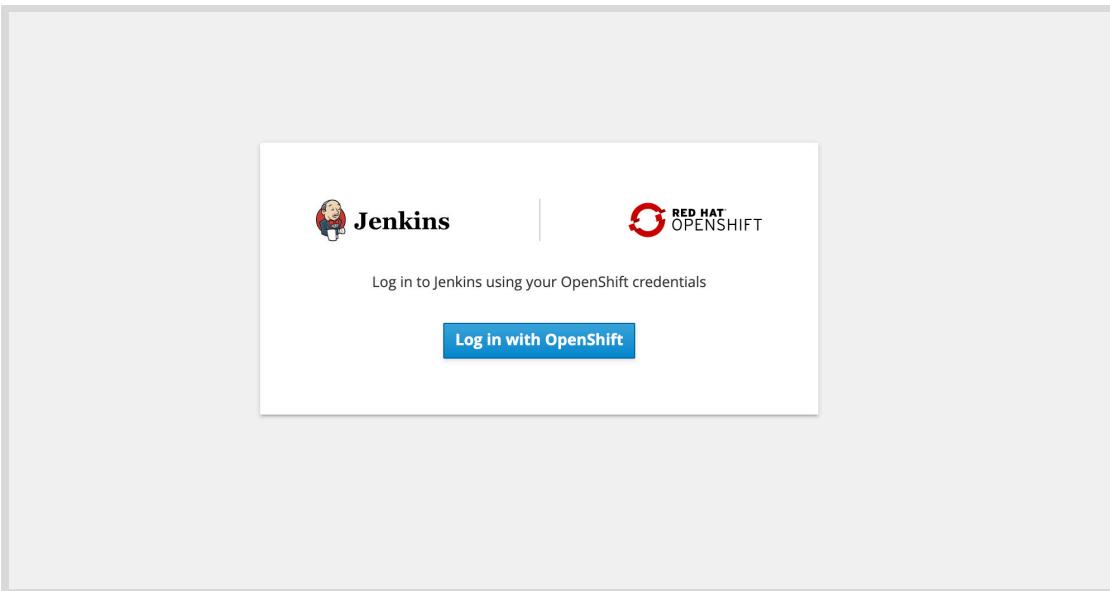


Figura 8.1: Inicio de sesión de Jenkins con credenciales de OpenShift

Una vez que inicie sesión, se le solicitará que autorice el acceso a su cuenta:

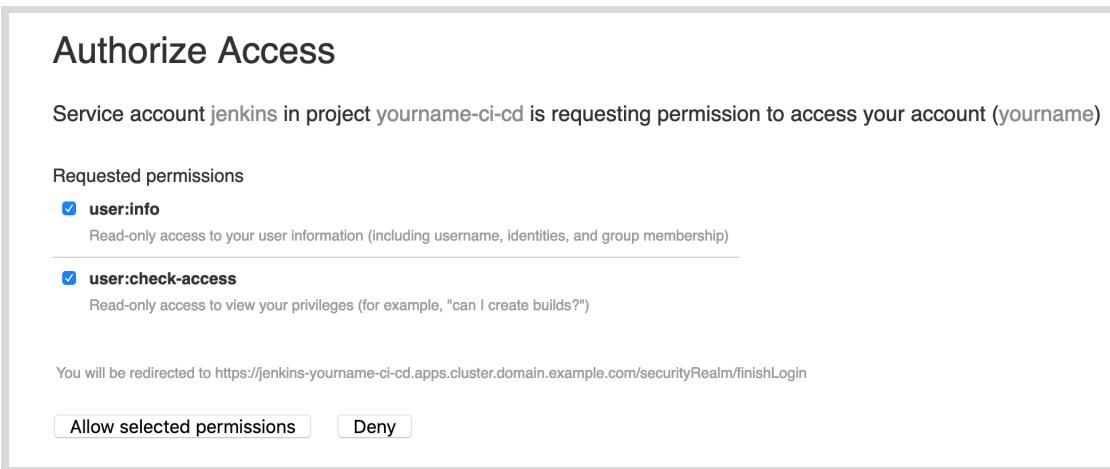


Figura 8.2: Autorizar el acceso a la cuenta de servicio

Permita que la cuenta de servicio acceda a los detalles de su cuenta para ver el panel de Jenkins.

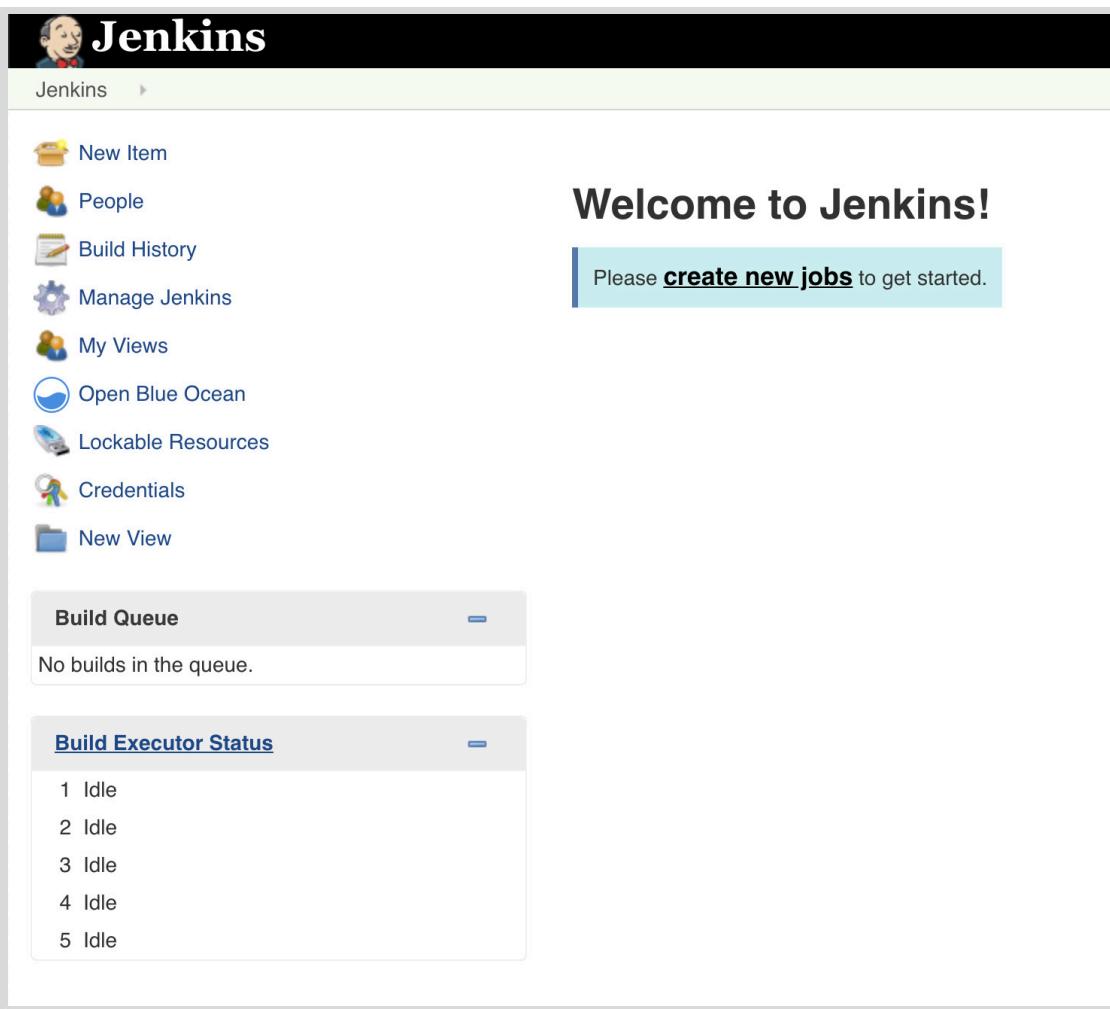


Figura 8.3: Página de inicio de Jenkins

6. Debe configurar el complemento de sincronización de Jenkins para monitorear el proyecto en busca de tuberías, así como para permitir que Jenkins interactúe con el clúster OpenShift en la creación y modificación de recursos.

Para configurar el complemento de sincronización de Jenkins, haga clic en **Manage Jenkins** (Administrar Jenkins) en el menú izquierdo de la página de inicio de Jenkins y, a continuación, haga clic en **Configure System** (Configurar sistema) en la página **Manage Jenkins** (Administrar Jenkins) para abrir la página de configuración global de Jenkins.

7. Desplácese hacia abajo hasta la sección **Sincronización de OpenShift Jenkins** y agregue el proyecto donde implementará la tubería en el campo **Namespace** (Espacio de nombres), junto al proyecto `myjenkins` ya existente. Separe las entradas con un espacio y no use una coma. El proyecto no necesita existir en OpenShift al realizar este paso. Creará el proyecto en pasos posteriores.

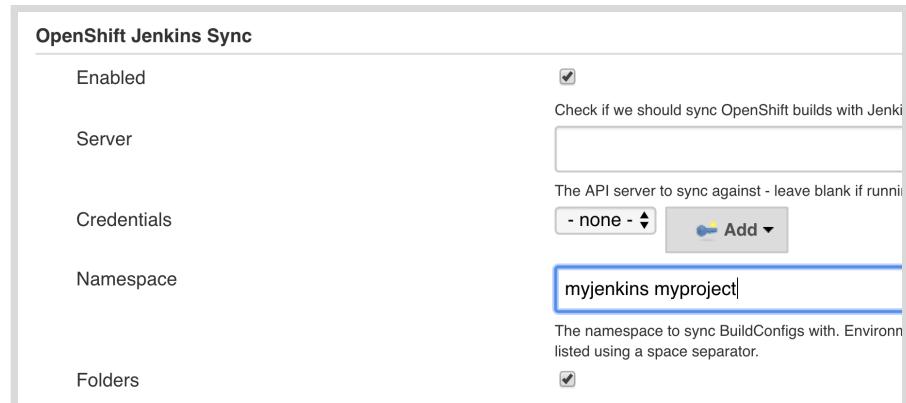


Figura 8.4: Configuración del complemento (plug-in) de sincronización de Jenkins

Haga clic en **Save** (Guardar) para aplicar los cambios a la configuración del complemento de sincronización de Jenkins.

- Cree un nuevo proyecto para implementar la aplicación. El nombre del proyecto debe estar en la lista de proyectos monitoreados por el cliente del complemento de sincronización de Jenkins.

```
[user@host ~]$ oc new-project myproject
```

- La cuenta de servicio asociada a la implementación de Jenkins debe tener el rol `edit` (edición) para cada proyecto monitoreado por Jenkins. Agregue el rol `edit` (edición) a la cuenta de servicio asociada con la implementación de Jenkins para su proyecto.

```
[user@host ~]$ oc policy add-role-to-user \
> edit system:serviceaccount:myjenkins:jenkins \
> -n myproject
```

- Cree de una configuración de compilación para la tubería. Una configuración de compilación de ejemplo tiene el siguiente aspecto:

```
kind: "BuildConfig"
apiVersion: "build.openshift.io/v1"
metadata:
  name: "mypipeline" ①
spec:
  source:
    contextDir: jenkins ②
    git:
      uri: "https://mygit/myapp" ③
      ref: "master"
  strategy:
    jenkinsPipelineStrategy: ④
    type: JenkinsPipeline ⑤
```

- ① El nombre de la configuración de compilación.
- ② El directorio relativo a la raíz del repositorio Git donde se almacena el archivo de Jenkins. Este atributo es opcional. OpenShift busca un archivo de Jenkins en la raíz del repositorio Git de forma predeterminada.

- ③ La dirección URL del repositorio Git donde se almacena el código fuente de la aplicación. El archivo de Jenkins suele guardarse en el mismo repositorio Git.
- ④ ⑤ La estrategia y el tipo de esta configuración de compilación, lo que indica que es una tubería de Jenkins.

Cree la configuración de la compilación:

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

11. Inicie una nueva compilación:

```
[user@host ~]$ oc start-build mypipeline
```

12. Una vez que se ejecuta la compilación, puede monitorear el progreso de la ejecución de la tubería desde la página **Builds (Compilaciones)** → **Builds** de la consola web de OpenShift.
13. Para ver detalles sobre la ejecución de la tubería, puede usar el panel de Jenkins para ver el registro de compilación de cada compilación. OpenShift crea una carpeta en el panel de Jenkins para cada proyecto donde se ejecutan las tuberías.



Referencias

Imagen de contenedor de OpenShift Jenkins

<https://github.com/openshift/jenkins>

Complemento (plug-in) de inicio de sesión de OpenShift Jenkins

<https://github.com/openshift/jenkins-openshift-login-plugin>

Complemento (plug-in) de sincronización de OpenShift Jenkins

<https://github.com/openshift/jenkins-sync-plugin>

Para obtener más información sobre la estrategia de compilación de tuberías, consulte el capítulo *Uso de estrategias de compilación* de la guía *Compilaciones* para Red Hat OpenShift Container Platform 4.5 en

https://docs.openshift.com/container-platform/4.5/builds/build-strategies.html#builds-strategy-pipeline-build_build-strategies

► Ejercicio Guiado

Ejecutar una tubería simple de Jenkins

En este ejercicio, implementará Jenkins para administrar tuberías en OpenShift y, a continuación, creará una tubería simple de Jenkins.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implementar una instancia de servidor de Jenkins en OpenShift.
- Crear y ejecutar una tubería simple de Jenkins mediante la estrategia de compilación de Pipeline (Tubería).

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La tubería de Jenkins de ejemplo en el repositorio Git (`simple-pipeline`).

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

```
[student@workstation ~]$ lab simple-pipeline start
```

► 1. Implemente una instancia de Jenkins en OpenShift.

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 OpenShift con su cuenta de usuario de desarrollador.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

1.3. Inspeccione las plantillas de Jenkins que están disponibles en OpenShift.

```
[student@workstation ~]$ oc get templates -n openshift | grep jenkins
jenkins-ephemeral    Jenkins service, without persistent storage....
jenkins-persistent    Jenkins service, with persistent storage....
```

Para este ejercicio, usará la plantilla `jenkins-ephemeral`.

- 1.4. Cree un proyecto nuevo para alojar la instancia de Jenkins. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-ci-cd
Now using project "youruser-ci-cd" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 1.5. Implemente una instancia de Jenkins.

```
[student@workstation ~]$ oc new-app --as-deployment-config \
> jenkins-ephemeral -p MEMORY_LIMIT=2048Mi
--> Deploying template "openshift/jenkins-ephemeral" to project youruser-ci-cd
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
...output omitted...
```

La instancia de Jenkins tardará un tiempo en implementarse. Avance a los pasos siguientes para inspeccionar y editar la configuración de compilación mientras Jenkins se está implementando.



nota

El parámetro `MEMORY_LIMIT` aumenta la cantidad de memoria asignada al contenedor de jenkins para mejorar el rendimiento. El tamaño máximo de la memoria dinámica de JVM se establece en el 50 % del valor de `MEMORY_LIMIT` de forma predeterminada. El valor predeterminado de `MEMORY_LIMIT` es 1 GB.

- 2. Edite la configuración de compilación para la tubería.

- 2.1. Inspeccione el archivo de recurso de configuración de compilación en `~/D0288/labs/simple-pipeline/simple-pipeline.yaml` con un editor de texto:

```
kind: "BuildConfig"
apiVersion: "build.openshift.io/v1"
metadata:
  name: "simple-pipeline" ①
spec:
  source:
    contextDir: simple-pipeline ②
    git:
      uri: "https://github.com/yourgituser/D0288-apps" ③
      ref: "simple-pipeline"
  strategy:
    jenkinsPipelineStrategy: ④
  type: JenkinsPipeline ⑤
```

- ① El nombre de la configuración de compilación.

- ❷ El directorio relativo a la raíz del repositorio Git donde se almacena el archivo de Jenkins.
- ❸ La dirección URL del repositorio Git donde se almacena el código fuente de la aplicación.
- ❹ ❺ La estrategia y el tipo de esta configuración de compilación, lo que indica que es una tubería de Jenkins.

2.2. Edite la dirección URL de Git donde se almacena el código fuente de la aplicación.

Edite la configuración de compilación y cambie el valor del atributo `spec.source.git.uri` para que apunte al repositorio Git:

```
git:
  uri: "https://github.com/yourgituser/D0288-apps"
  ref: "simple-pipeline"
```

► 3. Inspecione la tubería de Jenkins.

3.1. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

3.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b simple-pipeline
Switched to a new branch 'simple-pipeline'
[student@workstation D0288-apps]$ git push -u origin simple-pipeline
...output omitted...
* [new branch]      simple-pipeline -> simple-pipeline
Branch simple-pipeline set up to track remote branch simple-pipeline from origin.
```

3.3. Inspecione el archivo de Jenkins.

Edite el archivo `~/D0288-apps/simple-pipeline/Jenkinsfile` con un editor de texto. El archivo de Jenkins define una tubería de 4 etapas muy básica. La primera etapa (stage 1) y la última (stage 3) están completas.

En los pasos siguientes, completará la tubería implementando las líneas marcadas como `TODO`: en el archivo.

► 4. Complete la tubería de Jenkins.

4.1. Edite el archivo `~/D0288-apps/simple-pipeline/Jenkinsfile` y realice los siguientes pasos. También puede copiar estas instrucciones del archivo `~/D0288/solutions/simple-pipeline/Jenkinsfile` provisto.

4.2. Agregue una etiqueta a la sección `node` (nodo) para permitir que la tubería se ejecute en un nodo con la etiqueta `master`. Localice la línea:

```
// TODO: run this simple pipeline on jenkins 'master' node
```

Reemplace el código con lo siguiente:

```
...output omitted...
agent {
  node {
    label 'master'
  }
}
...output omitted...
```

- 4.3. Agregue una nueva etapa llamada `stage 2` después de `stage 1`. La nueva etapa simplemente imprime un mensaje de saludo. Agregue las siguientes líneas:

```
...output omitted...
stage('stage 2') {
  steps {
    sh 'echo hello from stage 2!'
  }
}
...output omitted...
```

- 4.4. Agregue una etapa después de `stage 2` que solicite la aprobación manual antes de ejecutar `stage 3`. Agregue las siguientes líneas:

```
...output omitted...
stage('manual approval') {
  steps {
    timeout(time: 60, unit: 'MINUTES') {
      input message: "Move to stage 3?"
    }
  }
}
...output omitted...
```

- 4.5. Guarde el archivo de Jenkins y confirme los cambios al repositorio Git de la carpeta `~/D0288-apps/simple-pipeline`:

```
[student@workstation D0288-apps]$ cd simple-pipeline
[student@workstation simple-pipeline]$ git commit -a -m "Completed Jenkinsfile"
...output omitted...
[student@workstation simple-pipeline]$ git push
...output omitted...
[student@workstation simple-pipeline]$ cd ~
```

- 5. Configure el complemento de sincronización de Jenkins y permita que Jenkins monitoree el proyecto en busca de tuberías.
- 5.1. Compruebe que la instancia de Jenkins que implementó anteriormente está lista y en ejecución:

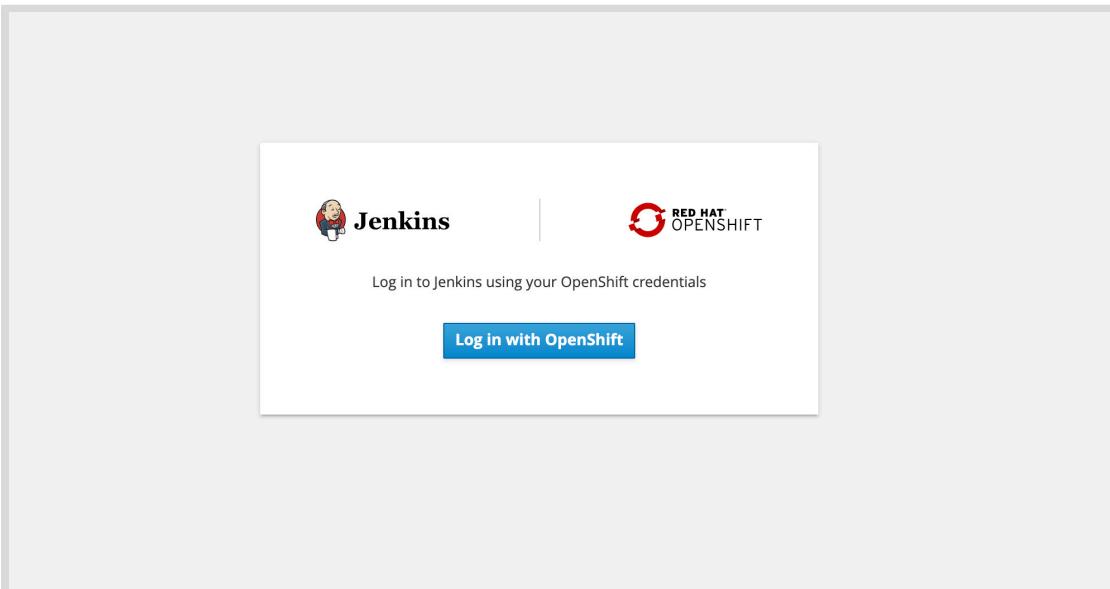
```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
jenkins-1-deploy  0/1     Completed  0          9m42s
jenkins-1-q5msj   1/1     Running   0          9m34s
```

- 5.2. Obtenga la dirección URL de ruta para la instancia de Jenkins:

```
[student@workstation ~]$ oc get route/jenkins -o jsonpath='{.spec.host}{"\n"}'
jenkins-youruser-ci-cd.apps.cluster.domain.example.com
```

- 5.3. Vaya a la dirección URL de ruta del paso anterior desde un navegador web.

Se le presentará una pantalla para iniciar sesión con sus credenciales de OpenShift.



Haga clic en **Log in with OpenShift** (Iniciar sesión con OpenShift) para abrir la página de inicio de sesión de la consola web de OpenShift.

- 5.4. Inicie sesión con su cuenta de usuario de desarrollador. Su nombre de usuario es la variable `RHT_OCP4_DEV_USER` del archivo de configuración del aula `/usr/local/etc/ocp4.config`. Su contraseña es la variable `RHT_OCP4_DEV_PASSWORD` del mismo archivo.
- 5.5. Se le mostrará una pantalla que le pedirá que autorice el acceso de la cuenta de servicio a su cuenta.

Authorize Access

Service account jenkins in project yourname-ci-cd is requesting permission to access your account (yourname)

Requested permissions

user:info

Read-only access to your user information (including username, identities, and group membership)

user:check-access

Read-only access to view your privileges (for example, "can I create builds?")

You will be redirected to <https://jenkins-yourname-ci-cd.apps.cluster.domain.example.com/securityRealm/finishLogin>

[Allow selected permissions](#)

[Deny](#)

Haga clic en **Allow selected permissions** (Permitir permisos seleccionados) para abrir la página de inicio de Jenkins.

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with the Jenkins logo and a dropdown menu. Below it is a sidebar with links: New Item, People, Build History, Manage Jenkins, My Views, Open Blue Ocean, Lockable Resources, Credentials, and New View. The main content area has a large "Welcome to Jenkins!" message with a call to action: "Please [create new jobs](#) to get started." Below this, there are two expandable sections: "Build Queue" (which shows "No builds in the queue.") and "Build Executor Status" (which lists 1, 2, 3, 4, and 5 Idle executors).

- 5.6. Haga clic en **Manage Jenkins** (Administrar Jenkins) en el menú izquierdo de la página de inicio de Jenkins y, a continuación, haga clic en **Configure System** (Configurar sistema) en la página **Manage Jenkins** (Administrar Jenkins) para abrir la página de configuración global de Jenkins.

- 5.7. Desplácese hacia abajo hasta la sección **Sincronización de OpenShift Jenkins** y agregue el espacio de nombres **youruser-simple-pipeline** al campo **Namespace** (Espacio de nombres), junto al espacio de nombres **youruser-ci-cd** ya existente. Separe las entradas con un espacio y no use una coma.

No cambie ningún otro parámetro en esta página. Creará el proyecto **youruser-simple-pipeline** en el paso siguiente.

Haga clic en **Save** (Guardar) para aplicar los cambios a la configuración global de Jenkins.

The screenshot shows the 'OpenShift Jenkins Sync' configuration page. It includes fields for 'Enabled' (checked), 'Server' (API server to sync against, set to 'none'), 'Credentials' (button to add), 'Namespace' (set to 'youruser-ci-cd youruser-simple-pipeline'), 'Folders' (checkbox checked), and 'Sync Job Name Pattern' (regular expression for matching pipeline job names).

► 6. Implemente la configuración de compilación en el proyecto e inicie una nueva compilación.

- 6.1. Cree un nuevo proyecto para ejecutar la tubería. Coloque como prefijo del nombre del proyecto el nombre de desarrollador.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-simple-pipeline
...output omitted...
```

Asegúrese de que el nombre del proyecto coincida con el espacio de nombres que agregó en la sección **Sincronización de OpenShift Jenkins** en el paso anterior.

- 6.2. La cuenta de servicio asociada a la implementación de Jenkins debe tener el rol **edit** (edición) para cada proyecto monitoreado por Jenkins. Agregue el rol **edit** (edición) a la cuenta de servicio asociada con la implementación de Jenkins para el proyecto **youruser-simple-pipeline**.

```
[student@workstation ~]$ oc policy add-role-to-user \
> edit system:serviceaccount:${RHT_OCP4_DEV_USER}-ci-cd:jenkins \
> -n ${RHT_OCP4_DEV_USER}-simple-pipeline
clusterrole.rbac.authorization.k8s.io/edit added:
"system:serviceaccount:youruser-ci-cd:jenkins"
```

- 6.3. Cree la configuración de la compilación:

```
[student@workstation ~]$ oc create \
> -f ~/D0288/labs/simple-pipeline/simple-pipeline.yaml
buildconfig.build.openshift.io/simple-pipeline created
```

- 6.4. Abra un navegador web y navegue en la consola web de OpenShift.

Inspeccione las rutas en el proyecto **openshift-console** para encontrar el nombre de host de su consola web de OpenShift.

```
[student@workstation ~]$ oc get route console -n openshift-console \
> -o jsonpath='{.spec.host}{"\n"}'
console-openshift-console.apps.cluster.domain.example.com
```

Inicie sesión con su cuenta de usuario de desarrollador.

- 6.5. Seleccione el proyecto **youruser-simple-pipeline**.

En la barra de navegación izquierda, haga clic en **Builds (Compilaciones)** → **Build Configs (Configuraciones de compilación)** y compruebe que una configuración de compilación denominada **simple-pipeline** esté visible.

- 6.6. Inicie una nueva compilación desde la línea de comandos para iniciar la tubería:

```
[student@workstation ~]$ oc start-build simple-pipeline
build.build.openshift.io/simple-pipeline-1 started
```

También puede usar la consola web de OpenShift para iniciar una compilación. Haga clic en la configuración de compilación **simple-pipeline** y, a continuación, haga clic en **Actions (Acciones)** → **Start Build (Iniciar compilación)** en la esquina superior derecha para iniciar una compilación.

- 7. Vea el progreso de ejecución de la tubería en la consola web de OpenShift.



nota

La estrategia de compilación de la tubería de Jenkins quedó obsoleta en OpenShift Container Platform 4. La funcionalidad equivalente está presente en las tuberías de OpenShift basadas en Tekton.

- 7.1. Una vez iniciada la compilación, vea el progreso de la tubería.

En la barra de navegación izquierda, haga clic en **Builds (Compilaciones)** → **Builds** y compruebe que una configuración de compilación denominada **simple-pipeline-1** esté visible.

- 7.2. Haga clic en **simple-pipeline-1** para ver la página de detalles de la compilación.

Después de un tiempo, debería ver las etapas de tubería ejecutándose. La ejecución se pausa en la etapa **manual approval** (aprobación manual).

Builds > Build Details

B simple-pipeline-1 Running

Actions ▾

Details YAML Environment Logs Events

Pipeline build strategy deprecation

With the release of [OpenShift Pipelines based on Tekton](#), the pipelines build strategy has been deprecated. Users should either use Jenkins files directly on Jenkins or use cloud-native CI/CD with Openshift Pipelines.

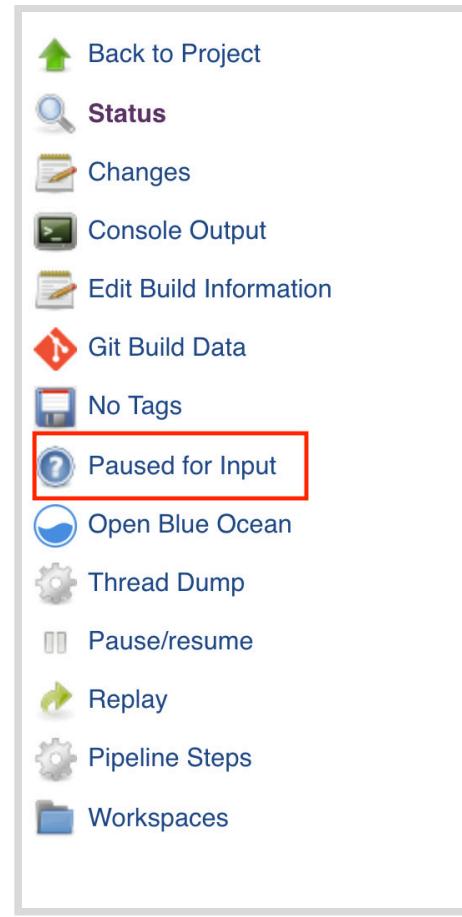
[Try the OpenShift Pipelines tutorial](#)

Build Details

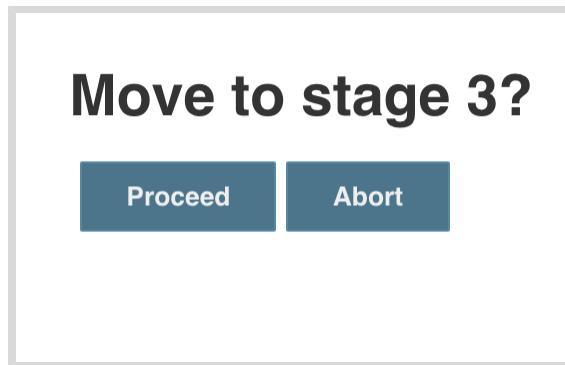
| | | | | |
|--|--|--|--|--|
|  Build 1 3 minutes ago View logs | Declarative....  2 minutes ago → | stage 1  2 minutes ago → | stage 2  2 minutes ago → | manual app...  Input Required Input Required 2 minutes ago |
|--|--|--|--|--|

7.3. Haga clic en **Input Required** (Entrada necesaria) debajo del ícono de pausa amarillo para abrir la página de detalles de compilación de Jenkins en una nueva pestaña del navegador.

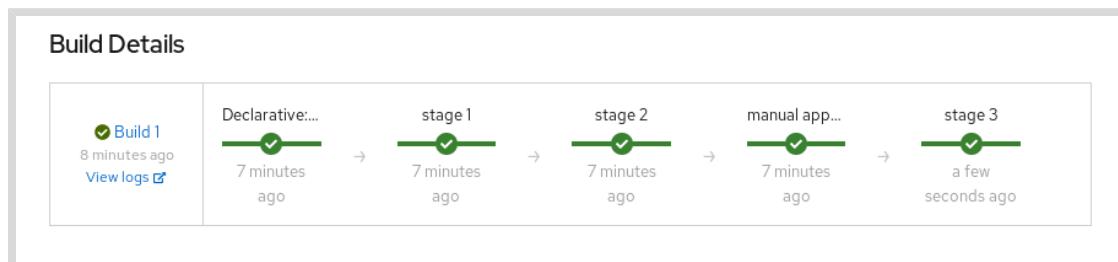
Haga clic en **Paused for Input** (Pausado para entrada) en el menú izquierdo de Jenkins cerca del ícono de signo de interrogación azul.



- 7.4. Haga clic en Proceed (Continuar) en el prompt Move to stage 3? (¿Pasar a etapa 3?).



- 7.5. La ejecución de la tubería continuará con stage 3 (etapa 3) y la tubería se completará correctamente.



► 8. Vea los detalles de la tubería ejecutada en la consola web de Jenkins.

- 8.1. Haga clic en el ícono **Jenkins** en la esquina superior izquierda de la consola de Jenkins para ver la página de inicio.

Ahora se muestra una nueva carpeta denominada *youruser-simple-pipeline*. Todos los registros de ejecución de tubería para este proyecto se almacenan dentro de esta carpeta. Haga clic en *youruser-simple-pipeline* para ver las tuberías de este proyecto.

- 8.2. Haga clic en *youruser-simple-pipeline/simple-pipeline* para ver el historial de compilación de la tubería. Haga clic en #1 (ícono de bola azul) en la sección **Build History** (Historial de compilación) del menú de la izquierda.

**nota**

Al hacer clic en el número de compilación, se abren los detalles de la compilación, mientras que al hacer clic en el ícono de la bola azul junto a este, se abre la página de salida de la consola para esa compilación.

Haga clic en **Console Output** (Salida de consola) para ver el registro de ejecución detallado de la tubería. Verá información de ejecución detallada sobre cada etapa de la tubería. La salida se ve de la siguiente manera:

```
...output omitted...
OpenShift Build youruser-simple-pipeline/simple-pipeline-1 from https://github.com/yourgituser/D0288-apps
...output omitted...
[Pipeline] Start of Pipeline
[Pipeline] node
...output omitted...
stage 1: using project: youruser-ci-cd in cluster https://172.30.0.1:443
...output omitted...
[Pipeline] sh
[workspace] Running shell script
+ echo hello from stage '2!'
...output omitted...
[Pipeline] input
Move to stage 3?
Proceed or Abort
Approved by youruser
...output omitted...
[Pipeline] { (stage 3)
[Pipeline] sh
[workspace] Running shell script
+ echo hello from stage '3!.' This is the last stage...
hello from stage 3!. This is the last stage...
...output omitted...
[Pipeline] End of Pipeline
Finished: SUCCESS
```

► 9. Limpieza: Elimine el proyecto *youruser-simple-pipeline*.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-simple-pipeline
```



Advertencia

NO elimine el proyecto `youruser-ci-cd`. Reutilizará la instancia de Jenkins en el siguiente ejercicio.

Finalizar

En `workstation`, ejecute el comando `lab simple-pipeline finish` para terminar este ejercicio. Este es un paso importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab simple-pipeline finish
```

Esto concluye el ejercicio guiado.

Escritura de tuberías de Jenkins personalizadas

Objetivos

Después de completar esta sección, debería poder crear y ejecutar una tubería de Jenkins personalizada.

Creación de una tubería de Jenkins personalizada

Puede crear una tubería de Jenkins personalizada para la aplicación escribiendo scripts en el lenguaje específico de dominio (DSL) de tubería de OpenShift en un archivo de Jenkins. El DSL de tubería de OpenShift tiene como objetivo proporcionar una sintaxis de tubería de Jenkins legible, concisa, completa y fluida que interactúa con un clúster OpenShift. El complemento (plug-in) proporciona funciones DSL de contenedor (wrapper) para la herramienta de línea de comandos de OpenShift (oc), que debe estar disponible en los nodos que ejecutan el script.

Elementos de un archivo de Jenkins

En la siguiente sección, se describen algunos de los elementos más importantes y usados con frecuencia de un archivo de Jenkins.

pipeline (tubería)

Un archivo de Jenkins comienza con el elemento de tubería declarado de la siguiente manera:

```
pipeline {
    ...output omitted...
}
```

options (opciones)

De manera opcional, puede declarar opciones globales que se aplican a toda la tubería en el elemento `options`:

```
pipeline {
    options {
        // set a timeout of 45 minutes for this pipeline
        timeout(time: 45, unit: 'MINUTES')
    }
    ...output omitted...
}
```

Las opciones se pueden invalidar por etapa.

agent (agente)

El elemento `agent` especifica el contexto de ejecución (donde se ejecutará la tubería) para toda la tubería. De manera opcional, puede invalidar el agente en un nivel por etapa. Use un elemento `label` (etiqueta) dentro del elemento `agent` para indicar flujos de imágenes con etiquetas específicas que se deben usar para ejecutar la tubería.

```

pipeline {
    options {
        ...output omitted...
    }
    agent {
        node {
            label 'master'
        }
    }
}

```

Un nodo con la etiqueta `master` está disponible de forma predeterminada en Jenkins. Contiene un tiempo de ejecución de Linux mínimo y se puede usar para ejecutar tuberías básicas mediante scripts de shell.

OpenShift proporciona un tiempo de ejecución basado en Apache Maven y OpenJDK para aplicaciones Java con la etiqueta `maven`. OpenShift también viene con un tiempo de ejecución de Node.js y NPM con la etiqueta `nodejs`.

Puede crear sus propias imágenes de contenedor personalizadas con etiquetas adecuadas y hacer que Jenkins las use para ejecutar la tubería. La creación de contenedores personalizados está fuera del ámbito de este curso. Consulte los recursos proporcionados en la sección de referencias para obtener más detalles.

environment (entorno)

Use el elemento `environment` para agregar variables y otras constantes que se usarán en toda la tubería. Este elemento está diseñado para capturar variables en un solo lugar a fin de evitar la duplicación de valores como direcciones URL, nombres de proyecto, nombres de aplicación y otras variables usadas en el código de tubería.

```

environment {
    DEV_PROJECT = "myapp-dev"
    STAGE_PROJECT = "myapp-stage"
    APP_GIT_URL = "https://mygitserver/myapp"
    NEXUS_SERVER = "http://mynexusserver/repository/java"
    ...output omitted...
}

```

stages (etapas) y steps (pasos)

Una tubería consta de una o varias `stages`. Cada stage representa un conjunto único de tareas que desea ejecutar en la tubería, como pruebas de unidad y de humo. Una stage consta de una serie de `steps` declarados dentro del elemento `steps`:

```

pipeline {
    options {
        ...output omitted...
    }
    ...output omitted...
    stages {
        stage('stage 1') {

```

```

        steps {
            ...output omitted...
        }
    }

    stage('stage 2') {
        steps {
            ...output omitted...
        }
    }

    stage('stage 3') {
        steps {
            ...output omitted...
        }
    }

}

```

script

El código DSL de tubería (código de lenguaje Groovy) debe estar incrustado dentro de los elementos `script`. Las funciones específicas de OpenShift se proporcionan en el espacio de nombres `openshift.*` y proporcionan funciones de contenedor (wrapper) en torno al cliente de línea de comandos `oc`.

A continuación se muestra un script de ejemplo que imprime el espacio de nombres y el clúster OpenShift actuales:

```

...output omitted...
stage('stage 1') {
    steps {
        script {
            openshift.withCluster() {
                openshift.withProject() {
                    echo "stage 1: using project:
${openshift.project()} in cluster ${openshift.cluster()}"
                }
            }
        }
    }
...output omitted...

```

Un ejemplo más complejo que inicia una nueva compilación es el siguiente:

```

...output omitted...
script {
    openshift.withCluster() {
        openshift.withProject(env.DEV_PROJECT) {
            openshift.selector("bc", "${APP_NAME}").startBuild("--wait=true", "--follow=true")
        }
    }

```

```

        }
    }
...output omitted...

```

sh (shell scripting)

Además de las tuberías de scripting que usan el DSL de tubería basado en Groovy dentro de elementos de `script`, puede incrustar scripts de shell para manipular el tiempo de ejecución de la tubería. La herramienta de línea de comandos `oc` está disponible en el tiempo de ejecución cuando se ejecuta la tubería.

Un script de shell de una línea simple se puede incrustar de la siguiente manera:

```

...output omitted...
stage('stage A') {
    steps {
        sh 'echo Hello World!'
    }
}

```

Un script de shell de varias líneas se puede incrustar de la siguiente manera:

```

...output omitted...
stage('stage A') {
    steps {
        sh''''
            echo 'deleting all resources...'
            oc project ${DEV_PROJECT}
            oc delete all -l app=${APP_NAME}
            sleep 5
        '''
    }
}

```



Advertencia

Al crear scripts de tubería mediante scripts de DSL y shell, asegúrese de que está en el proyecto correcto en cada etapa antes de ejecutar comandos. El contexto del proyecto actual cambia al proyecto predeterminado entre etapas.

Cuando se usan scripts de shell, es siempre una buena idea usar la opción `-n` para definir explícitamente el proyecto al ejecutar comandos `oc`. Del mismo modo, para el código DSL, haga un uso eficaz de las funciones `openshift.withCluster()` y `openshift.withProject()` antes de manipular los recursos en OpenShift.

input (entrada)

Use el elemento `input` (entrada) para agregar interacción manual a la tubería. Esto se usa con mayor frecuencia en escenarios en los que se promociona la aplicación entre entornos y se necesita una aprobación manual para que la tubería continúe.

Por ejemplo, para asegurarse de que las promociones desde el desarrollo hasta la puesta en escena requieran siempre aprobación manual, use lo siguiente:

```
...output omitted...
stage('Promote to Staging Env') {
    steps {
        timeout(time: 60, unit: 'MINUTES') {
            input message: "Promote to Staging?"
        }
        script {
            // code to promote app
        }
    }
}
...output omitted...
```

El elemento *timeout* (tiempo de espera) garantiza que si no se proporciona la aprobación manual después de 60 minutos, la ejecución de la tubería continúa.



Referencias

DSL de tubería de complementos de cliente de Jenkins

<https://github.com/openshift/jenkins-client-plugin>

Sintaxis de DSL de tubería de Jenkins

<https://jenkins.io/doc/book/pipeline/syntax/>

► Ejercicio Guiado

Crear y ejecutar una tubería de Jenkins

En este ejercicio, creará y ejecutará una tubería de Jenkins para implementar un microservicio de Node.js.

Resultados

Deberá ser capaz de crear y ejecutar una tubería de Jenkins de varias etapas en OpenShift para implementar un microservicio de Node.js.

Andes De Comenzar



Importante

Debe completar los pasos del 1.1 al 1.5 del ejercicio anterior (*Ejecutar una tubería simple de Jenkins*) antes de intentar resolver este ejercicio.

Para realizar este ejercicio, debe tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Una instancia de Jenkins implementada y en ejecución en el espacio de nombres `youruser-ci-cd`.
- La aplicación de muestra del repositorio Git (`books`).

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos de soluciones:

```
[student@workstation ~]$ lab custom-pipeline start
```

- 1. Compruebe que la instancia de Jenkins que implementó en el ejercicio anterior esté lista y en ejecución.

- 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 OpenShift con su cuenta de usuario de desarrollador.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

13. Compruebe que el pod de Jenkins esté en ejecución:

```
[student@workstation ~]$ oc get pods -n ${RHT_OCP4_DEV_USER}-ci-cd
NAME          READY   STATUS    RESTARTS   AGE
...output omitted...
jenkins-1-q5msj   1/1     Running   0          9m34s
```

- 2. Cree dos proyectos para los entornos de desarrollo y ensayo para el microservicio. La tubería implementará primero el microservicio books (libros) en el entorno de desarrollo y, a continuación, lo promoverá al entorno de ensayo después de ejecutar pruebas de unidad y herramientas de Lint en el código.

- 2.1. Cree un nuevo proyecto para el entorno de ensayo. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-books-stage
Now using project "youruser-books-stage" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 2.2. Cree un nuevo proyecto para el entorno de desarrollo. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-books-dev
Now using project "youruser-books-dev" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 3. Edite la configuración de compilación para la tubería.

- 3.1. Inspeccione el archivo de recurso de configuración de compilación de tubería en ~/D0288/labs/custom-pipeline/custom-pipeline-bc.json con un editor de texto:

```
{
  "kind": "BuildConfig",
  "apiVersion": "build.openshift.io/v1",
  "metadata": {
    "name": "custom-pipeline" ①
  },
  "spec": {
    "source": {
      "type": "Git",
      "git": {
        "uri": "https://github.com/youruser/D0288-apps.git", ②
        "ref": "custom-pipeline" ③
      },
      "contextDir": "books" ④
    },
    "strategy": {
      "type": "JenkinsPipeline",
      "jenkinsPipelineStrategy": {
        "jenkinsfilePath": "jenkins/Jenkinsfile" ⑤
      }
    }
  }
}
```

```

        }
    }
}
}
```

- 1** El nombre de la configuración de compilación.
- 2** La dirección URL del repositorio Git donde se almacena el archivo de Jenkins.
- 3** El nombre de bifurcación (en el repositorio Git) que se usará para la compilación.
- 4** El nombre del directorio bajo el cual se guarda la fuente de la aplicación.
- 5** El nombre del directorio relativo a `contextDir` donde se guarda el archivo de Jenkins.

**nota**

En este ejercicio, almacenamos la fuente de la aplicación para la aplicación books y el archivo de Jenkins en el mismo repositorio Git. No es obligatorio hacer esto, pero es una práctica recomendada.

- 3.2. Edite la dirección URL de Git donde se almacena el código fuente de la aplicación.

Edite la configuración de compilación y cambie el valor del atributo `spec.source.git.uri` para que apunte al repositorio Git:

```

...output omitted...
git: {
  uri: "https://github.com/youruser/D0288-apps",
  ref: "custom-pipeline"
},
...output omitted...
```

► **4.** Inspeccione la tubería de Jenkins.

- 4.1. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 4.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b custom-pipeline
Switched to a new branch 'custom-pipeline'
[student@workstation D0288-apps]$ git push -u origin custom-pipeline
...output omitted...
* [new branch]      custom-pipeline -> custom-pipeline
Branch custom-pipeline set up to track remote branch custom-pipeline from origin.
```

- 4.3. Inspeccione el archivo de Jenkins en `~/D0288-apps/books/jenkins/Jenkinsfile`. El archivo de Jenkins define una tubería de varias etapas para

implementar el microservicio. Algunas de las etapas están completas, mientras que otras están incompletas.

En los pasos siguientes, revise y complete la tubería implementando las líneas marcadas como **TODO**: en el archivo.

► 5. Revise las etapas de tubería y complete la tubería de Jenkins.

- 5.1. Edite el archivo `~/D0288-apps/books/jenkins/Jenkinsfile` y realice los siguientes pasos. También puede copiar las etapas completadas del archivo `~/D0288/solutions/custom-pipeline/Jenkinsfile` provisto.
- 5.2. La tubería debe ejecutarse en un nodo de agente de Jenkins que admite Node.js. Agregue una etiqueta a la sección `node` (nodo) para permitir que la tubería se ejecute en un nodo con la etiqueta `nodejs`. Localice la línea:

```
//TODO: Add label for Node.js jenkins agent
```

Reemplace el código con lo siguiente:

```
...output omitted...
agent {
  node {
    label 'nodejs'
  }
}
...output omitted...
```

- 5.3. Personalice la sección `environment` (entorno) y agregue valores según su entorno.

Reemplace el valor de `youruser` por su cuenta de usuario de desarrollador.

Reemplace el valor del atributo `NEXUS_SERVER` por el valor de la variable `RHT_OCP4_NEXUS_SERVER` del archivo `/usr/local/etc/ocp4.config`.

```
...output omitted...
environment {
  //TODO: Edit these vars as per your env
  DEV_PROJECT = "youruser-books-dev"
  STAGE_PROJECT = "youruser-books-stage"
  APP_GIT_URL = "https://github.com/youruser/D0288-apps"
  NEXUS_SERVER = "http://nexus-common.apps.cluster.domain.example.com/repository/
nodejs"

  // DO NOT CHANGE THE GLOBAL VARS BELOW THIS LINE
  APP_NAME = "books"
}
...output omitted...
```

- 5.4. Las etapas `NPM Install` (Instalar NPM), `Run Unit Tests` (Ejecutar pruebas de unidad) y `Run Linting Tools` (Ejecutar herramientas de Lint) instalan las dependencias del microservicio, ejecutan pruebas de unidad y ejecutan Lint en el código, respectivamente.

Estas etapas están diseñadas para detectar errores al principio de la etapa de desarrollo y proporcionan comentarios rápidos a los desarrolladores siguiendo un enfoque de "error rápido". No realice ningún cambio a estas etapas.

- 5.5. Edite la etapa **Launch new app in DEV env** (Iniciar aplicación nueva en entorno de desarrollo) e inicie una nueva aplicación de OpenShift en el entorno de desarrollo.

Localice la línea:

```
//TODO: Create a new app and expose the service
```

Reemplácela con las siguientes líneas:

```
...output omitted...
sh '''
    oc project ${DEV_PROJECT}
    oc new-app --as-deployment-config --name books nodejs:12~${APP_GIT_URL} \
    --build-env npm_config_registry=${NEXUS_SERVER} \
    --context-dir ${APP_NAME}

    oc expose svc/${APP_NAME}
'''

...output omitted...
```

- 5.6. Las etapas **Wait for S2I build to complete** (Esperar a que termine la compilación S2I) y **Wait for deployment in DEV env** (Esperar la implementación en el entorno de desarrollo) para que la compilación y la implementación del microservicio se completen correctamente.

Tenga en cuenta el uso del complemento DSL de tubería de OpenShift en estas etapas, en lugar de los comandos de script de shell sin procesar, para interactuar con la API maestra de OpenShift. No realice ningún cambio en estas etapas.

- 5.7. Edite la etapa **Promote to Staging Env** (Promover a entorno de ensayo) y etiquete el flujo de imágenes más reciente del microservicio.

Localice la línea:

```
//TODO: Tag the books:latest image stream as books:stage
```

Reemplácela con las siguientes líneas:

```
...output omitted...
script {
    openshift.withCluster() {
        openshift.tag("${DEV_PROJECT}/books:latest", "${STAGE_PROJECT}/books:stage")
    }
}
...output omitted...
```

- 5.8. Edite la etapa **Deploy to Staging Env** (Implementar en entorno de ensayo) y agregue código para crear una nueva aplicación con el flujo de imágenes etiquetado en la etapa anterior.

Localice la línea:

```
//TODO: Create a new app in stage using the books:stage image stream and expose the service
```

Reemplácela con las siguientes líneas:

```
...output omitted...
sh ''
  oc project ${STAGE_PROJECT}
  oc new-app --as-deployment-config --name books -i books:stage
  oc expose svc/${APP_NAME}
...
...output omitted...
```

- 5.9. Edite la etapa **Wait for deployment in Staging** (Esperar la implementación en la etapa de ensayo) y agregue código para comprobar que los pods de la aplicación se están ejecutando. Use el código de la etapa **Wait for deployment in DEV env** (Esperar la implementación en el entorno de desarrollo) como referencia para completar esta etapa.

Localice la línea:

```
//TODO: Watch deployment until pod is in 'Running' state
```

Reemplácela con las siguientes líneas:

```
...output omitted...
openshift.withProject( "${STAGE_PROJECT}" ) {
  def deployment = openshift.selector("dc", "${APP_NAME}").rollout()
  openshift.selector("dc", "${APP_NAME}").related('pods').untilEach(1) {
    return (it.object().status.phase == "Running")
  }
}
...output omitted...
```

El archivo de Jenkins ahora está completo. Guarde los cambios.

- 5.10. Confirme los cambios al repositorio Git desde la carpeta ~/D0288-apps/books:

```
[student@workstation D0288-apps]$ cd ~/D0288-apps/books
[student@workstation books]$ git commit -a \
> -m "Completed Jenkinsfile for books microservice"
...output omitted...
[student@workstation books]$ git push
...output omitted...
[student@workstation books]$ cd ~
```

- 6. Configure el complemento de sincronización de Jenkins y permita que Jenkins monitoree el proyecto en busca de tuberías.

- 6.1. Obtenga la dirección URL de ruta para la instancia de Jenkins:

```
[student@workstation ~]$ oc get route/jenkins -n ${RHT_OCP4_DEV_USER}-ci-cd \
> -o jsonpath='{.spec.host}{"\n"}'
jenkins-youruser-ci-cd.apps.cluster.domain.example.com
```

- 6.2. Vaya a la dirección URL de ruta del paso anterior desde un navegador web.

Se le presentará una pantalla para iniciar sesión con sus credenciales de OpenShift. Haga clic en **Log in with OpenShift** (Iniciar sesión con OpenShift) para abrir la página de inicio de sesión de la consola web de OpenShift.

- 6.3. Inicie sesión con su cuenta de usuario de desarrollador. Su nombre de usuario es la variable `RHT_OCP4_DEV_USER` del archivo de configuración del aula `/usr/local/etc/ocp4.config`. Su contraseña es la variable `RHT_OCP4_DEV_PASSWORD` del mismo archivo.
- 6.4. Si accede a la consola de Jenkins por primera vez, se muestra una pantalla que le pide que autorice el acceso de la cuenta de servicio a su cuenta. Haga clic en **Allow selected permissions** (Permitir permisos seleccionados) para abrir la página de inicio de Jenkins.
- 6.5. Haga clic en **Manage Jenkins** (Administrar Jenkins) en el menú izquierdo de la página de inicio de Jenkins y, a continuación, haga clic en **Configure System** (Configurar sistema) en la página **Manage Jenkins** (Administrar Jenkins) para abrir la página de configuración global de Jenkins.
- 6.6. Desplácese hacia abajo hasta la sección **OpenShift Jenkins Sync** (Sincronización de OpenShift Jenkins) y agregue el espacio de nombres `youruser-books-dev` al campo **Namespace** (Espacio de nombres), junto a los espacios de nombres `youruser-ci-cd` y `youruser-simple-pipeline` ya existentes. Separe las entradas con un espacio y no use una coma.

No cambie ningún otro parámetro en esta página. Haga clic en **Save** (Guardar) para aplicar los cambios a la configuración global de Jenkins.

| OpenShift Jenkins Sync | |
|------------------------|---|
| Enabled | <input checked="" type="checkbox"/> Check if we should sync OpenShift builds with Jenkins jobs |
| Server | |
| Credentials | - none - <input type="button" value="Add"/> |
| Namespace | youruser-ci-cd youruser-simple-pipeline youruser-books-dev |
| Folders | <input checked="" type="checkbox"/> |

► 7. Implemente la configuración de compilación en el proyecto e inicie una nueva compilación.

- 7.1. Asegúrese de crear la configuración de compilación en el espacio de nombres `youruser-books-dev`.

```
[student@workstation ~]$ oc project ${RHT_OCP4_DEV_USER}-books-dev
Now using project "youruser-books-dev" on server
...output omitted...
```

- 7.2. La cuenta de servicio asociada a la implementación de Jenkins debe tener el rol `edit` (edición) para cada proyecto en el que Jenkins realiza alguna operación. Agregue el rol `edit` (edición) a la cuenta de servicio asociada con la implementación de Jenkins para los proyectos `youruser-books-dev` y `youruser-books-stage`.

```
[student@workstation ~]$ oc policy add-role-to-user \
> edit system:serviceaccount:${RHT_OCP4_DEV_USER}-ci-cd:jenkins \
> -n ${RHT_OCP4_DEV_USER}-books-dev
clusterrole.rbac.authorization.k8s.io/edit added:
"system:serviceaccount:youruser-ci-cd:jenkins"

[student@workstation ~]$ oc policy add-role-to-user \
> edit system:serviceaccount:${RHT_OCP4_DEV_USER}-ci-cd:jenkins \
> -n ${RHT_OCP4_DEV_USER}-books-stage
clusterrole.rbac.authorization.k8s.io/edit added:
"system:serviceaccount:youruser-ci-cd:jenkins"
```

7.3. Cree la configuración de la compilación:

```
[student@workstation ~]$ oc create \
> -f ~/D0288/labs/custom-pipeline/custom-pipeline-bc.json
buildconfig.build.openshift.io/custom-pipeline created
```

7.4. Abra un navegador web y navegue a la consola web de OpenShift.

Inspeccione las rutas en el proyecto `openshift-console` para encontrar el nombre de host de su consola web de OpenShift.

```
[student@workstation ~]$ oc get route console -n openshift-console \
> -o jsonpath='{.spec.host}{ "\n" }'
console-openshift-console.apps.cluster.domain.example.com
```

Inicie sesión con su cuenta de usuario de desarrollador.



nota

La estrategia de compilación de la tubería de Jenkins quedó obsoleta en OpenShift Container Platform 4. La funcionalidad equivalente está presente en las tuberías de OpenShift basadas en Tekton.

7.5. Seleccione el proyecto `youruser-books-dev`.

En la barra de navegación izquierda, haga clic en **Builds (Compilaciones)** → **Build Configs (Configuraciones de compilación)** y compruebe que una configuración de compilación denominada `custom-pipeline` esté visible.

7.6. Inicie una nueva compilación desde la línea de comandos para iniciar la tubería:

```
[student@workstation ~]$ oc start-build custom-pipeline
build.build.openshift.io/custom-pipeline-1 started
```

También puede usar la consola web de OpenShift para iniciar una compilación. Haga clic en la configuración de compilación `custom-pipeline` y, a continuación, haga clic en **Actions (Acciones)** → **Start Build (Iniciar compilación)** en la esquina superior derecha para iniciar una compilación.

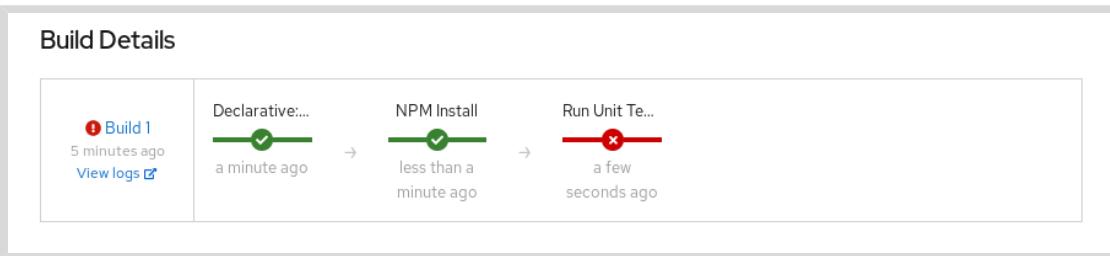
► 8. Vea el progreso de la tubería.

8.1. Una vez iniciada la compilación, vea el progreso de la tubería.

En la barra de navegación izquierda, haga clic en **Builds (Compilaciones) → Builds** y compruebe que una compilación denominada **custom-pipeline-1** esté visible.

- 8.2. Haga clic en **custom-pipeline-1** para ver la página de detalles de la compilación.

Después de un tiempo, debería ver las etapas de tubería ejecutándose. Se produce un error en la ejecución en la etapa **Run Unit Tests** (Ejecutar pruebas de unidad).



- ▶ 9. Corrija las pruebas de unidad con errores y vuelva a ejecutar la tubería.
- 9.1. En la página **Build Details** (Detalles de compilación), haga clic en **View Logs** (Ver registros) (debajo de **Build 1** [Compilación 1] con el signo de exclamación en una burbuja roja) para ver el registro de ejecución de tubería de Jenkins.
 - 9.2. En el registro de ejecución de tubería de Jenkins, se muestra un error en la etapa **Run Unit Tests** (Ejecutar pruebas de unidad):

```
...output omitted...
+ cd books
+ npm test

> books@1.0.0 test /tmp/workspace/ ...output omitted...
> IP=0.0.0.0 PORT=3030 node_modules/.bin/mocha tests/*_test.js
...output omitted...
2 passing (138ms)
  1 failing

  1) Books App routes test
     GET to /authors should return 200:
       Uncaught AssertionError: expected '{"books":[{"id":1,"name":"James Joyce","dob":1882},{"id":2,"name":"F Scott Fitzgerald","dob":1896},{"id":3,"name":"Aldous Huxley","dob":1894},{"id":4,"name":"Vladimir Nabokov","dob":1899},{"id":5,"name":"William Faulkner","dob":1897}]}' to include 'James_Joyce'
     ...output omitted...
```

La llamada HTTP GET al extremo (endpoint) `/authors` devuelve una matriz de objetos de autor. La prueba comprueba si la cadena "James_Joyce" está presente en la salida. La prueba falla porque la respuesta incluye la cadena "James Joyce".

- 9.3. Corrija la prueba de unidad que falla. Abra el archivo `~/DO288-apps/books/tests/app_test.js` en un editor de texto.
- 9.4. Busque la siguiente línea en el caso de prueba '`GET to /authors should return 200`' (`GET a /authors` debe devolver 200) en la parte inferior:

```
expect(res.text).to.include('James_Joyce');
```

Reemplace el código con la siguiente afirmación correcta:

```
expect(res.text).to.include('James Joyce');
```

- 9.5. Confirme los cambios y envíe el archivo de prueba fijo a la bifurcación `youruser-custom-pipeline`.

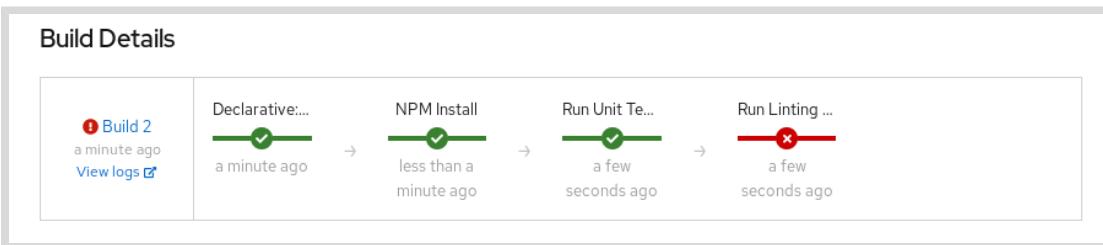
```
[student@workstation ~]$ cd ~/D0288-apps/books
[student@workstation books]$ git commit -a -m "Fixed failed unit test."
...output omitted...
[student@workstation books]$ git push
...output omitted...
[student@workstation books]$ cd ~
```

- 9.6. Inicie una nueva compilación para volver a ejecutar la tubería.

```
[student@workstation ~]$ oc start-build custom-pipeline
build.build.openshift.io/custom-pipeline-2 started
```

- 9.7. Monitoree el progreso de la ejecución de la tubería para la compilación `custom-pipeline-2` desde la página **Build Details** (Detalles de compilación).

La tubería ahora falla en la etapa **Run Linting Tools** (Ejecutar herramientas de Lint).



Compruebe el registro de ejecución de tubería de Jenkins con un clic en **View Logs** (Ver registros). Los registros muestran que la herramienta de Lint se queja de variables no usadas en el código, así como del uso de la palabra clave `var`, que no se recomienda en las versiones más recientes de JavaScript.

```
...output omitted...
+ cd books
+ npm run lint

> books@1.0.0 lint /tmp/workspace/ ...output omitted...
> eslint . --ext .js

...output omitted.../books/routes/authors.js
  7:1  error  Unexpected var, use let or const instead  no-var
  7:5  error  'user' is defined but never used          no-unused-vars
...output omitted...
```

- 10. Corrija los errores de Lint y vuelva a ejecutar la tubería.

- 10.1. Abra el archivo `~/D0288-apps/books/routes/authors.js` en un editor de texto.

10.2. Elimine la siguiente línea de código:

```
var user;
```

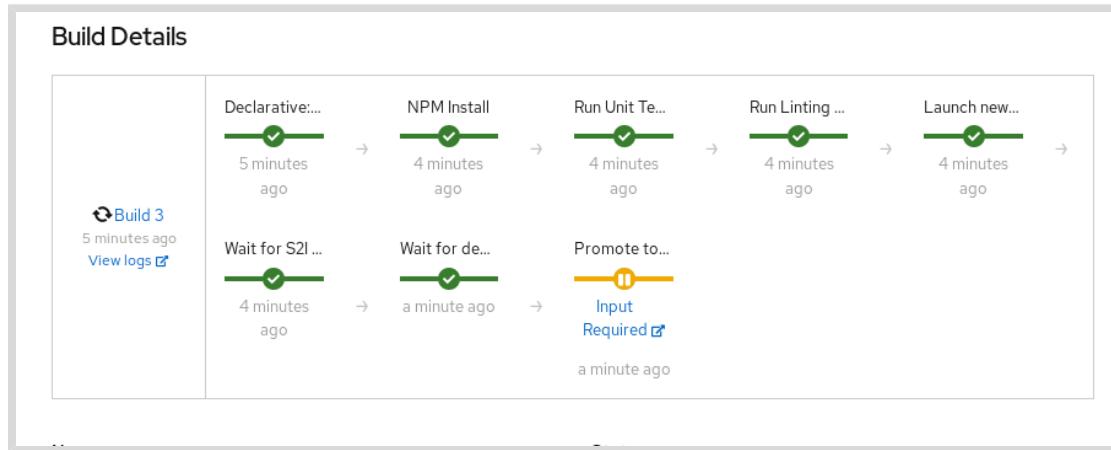
10.3. Confirme los cambios y envíe el archivo de prueba fijo a la bifurcación `youruser-custom-pipeline`.

```
[student@workstation ~]$ cd ~/DO288-apps/books
[student@workstation books]$ git commit -a -m "Fixed linting errors."
...output omitted...
[student@workstation books]$ git push
...output omitted...
[student@workstation books]$ cd ~
```

10.4. Inicie una nueva compilación para volver a ejecutar la tubería.

```
[student@workstation ~]$ oc start-build custom-pipeline
build.build.openshift.io/custom-pipeline-3 started
```

Monitoree el progreso de la ejecución de la tubería para la compilación `custom-pipeline-3` desde la página **Build Details** (Detalles de compilación). La tubería ahora debe continuar con la creación y el lanzamiento del microservicio en el entorno de desarrollo. Una vez completadas estas etapas, la ejecución de la tubería se pausa en la etapa **Promote to Staging Env** (Promover a entorno de ensayo).



► 11. Compruebe que el microservicio esté implementado y en ejecución en el entorno de desarrollo.

11.1. Antes de promover el microservicio al entorno de ensayo, compruebe que la aplicación se esté ejecutando en el entorno de desarrollo.

Verifique que el microservicio `books` se esté ejecutando:

```
[student@workstation ~]$ oc get pods -n ${RHT_OCP4_DEV_USER}-books-dev
NAME          READY   STATUS    RESTARTS   AGE
...output omitted...
books-1-b6h94  1/1     Running   0          17m
```

11.2. Compruebe que el entorno de ensayo no tenga ningún pod en ejecución:

```
[student@workstation ~]$ oc get pods -n ${RHT_OCP4_DEV_USER}-books-stage  
No resources found.
```

- 11.3. Obtenga la dirección URL de ruta del microservicio books:

```
[student@workstation ~]$ oc get route/books \  
> -n ${RHT_OCP4_DEV_USER}-books-dev \  
> -o jsonpath='{.spec.host}{\"\n\"}'  
books-youruser-books-dev.apps.cluster.domain.example.com
```

- 11.4. Vaya a la dirección URL de ruta del paso anterior desde un navegador web y compruebe que el microservicio books esté en ejecución.

The Book List

Welcome to The Book List

The List of books is [here](#).

The List of authors is [here](#).

- 12. Promueva el microservicio al entorno de ensayo.

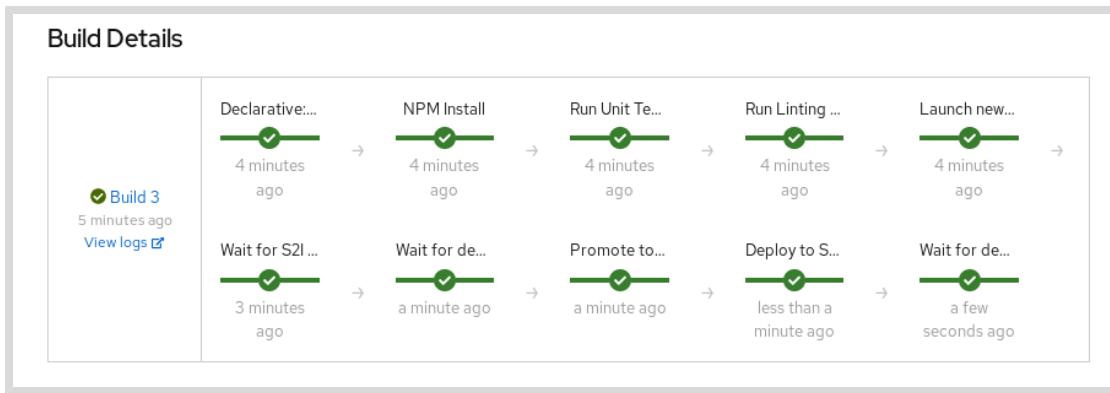
- 12.1. Una vez que esté satisfecho de que el microservicio funciona según lo previsto, promueva el microservicio al entorno de ensayo.
- 12.2. En la página **Build Details** (Detalles de compilación) para la compilación **custom-pipeline-3**, haga clic en **Input Required** (Entrada requerida) para ver el registro de ejecución de tubería de Jenkins.
- 12.3. En el menú izquierdo de la consola de Jenkins, haga clic en **Paused for Input** (Pausado para entrada). Haga clic en **Proceed** (Continuar) en el prompt **Promote to Staging?** (¿Promover al ensayo?).
- 12.4. Jenkins cambia a la ventana de registro de ejecución y continúa ejecutando la etapa **Deploy to Staging Env** (Implementar en entorno de ensayo).

- 13. Compruebe que la ejecución de la tubería se realice correctamente. Compruebe que el microservicio se implemente y esté en ejecución en el entorno de ensayo.

- 13.1. La ejecución de la tubería debe completarse y debería ver lo siguiente en el registro de ejecución:

```
...output omitted...
Deployment to Staging env is complete.
Access the app at the URL
http://books-youruser-books-stage.apps.cluster.domain.example.com
...output omitted...
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- 13.2. Haga clic en la dirección URL del microservicio books en el entorno de ensayo.
Compruebe que el microservicio sea similar en funciones a la versión implementada en el entorno de desarrollo.
- 13.3. En la página Build Details (Detalles de compilación) para **custom-pipeline-3**, se muestra el estado completo de ejecución de la tubería.



- 14. Limpieza: Elimine los proyectos *youruser-ci-cd*, *youruser-books-dev* y *youruser-books-stage*.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-books-dev
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-books-stage
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-ci-cd
```

Finalizar

En **workstation**, ejecute el comando **lab custom-pipeline finish** para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab custom-pipeline finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Implementación de canalizaciones de integración continua y de implementación continua en OpenShift

Listado de verificación de rendimiento

En este trabajo de laboratorio, implementará Jenkins y creará una tubería de CI/CD para implementar un microservicio Java en OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implementar una instancia efímera de Jenkins en OpenShift.
- Crear un archivo de Jenkins con varias etapas para implementar el microservicio en el entorno de desarrollo.
- Promover el microservicio del entorno de desarrollo al entorno de ensayo.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación de muestra (movies) en el repositorio Git.
- La imagen del compilador S2I OpenJDK 8 requerida por la aplicación.

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de proyecto inicial y los archivos de soluciones:

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

Requisitos

El microservicio (movies) proporciona una API REST que enumera las películas. Está escrito en Java con el marco (framework) Spring Boot.

El microservicio se implementa de un código fuente almacenado en un repositorio Git.

Implementará una instancia efímera de Jenkins en OpenShift y la configurará para ejecutar tuberías desde espacios de nombres seleccionados en OpenShift.

Un archivo de recurso de configuración de compilación de inicio denominado `movies-bc.json` se ofrece en la carpeta `~/DO288/labs/review-cicd`. Debe editar este archivo y personalizarlo para su entorno.

En el repositorio Git se proporciona un archivo de Jenkins de inicio que contiene las etapas de tubería junto con el código fuente de la aplicación. Debe completar el archivo de Jenkins e implementarlo en OpenShift.

Una vez que implemente la tubería y la ejecute, verá errores en determinadas etapas. Corrija estos errores analizando el registro de ejecución de tubería de Jenkins y haga que la tubería ejecute todas las etapas correctamente.

Implemente el microservicio y la tubería de Jenkins en conformidad con los siguientes requisitos:

- Implemente una instancia efímera de Jenkins en un espacio de nombres denominado *youruser-jenkins*. Asegúrese de que el parámetro **MEMORY_LIMIT** de la plantilla está establecido en 2GB para mejorar el rendimiento.
- El nuevo proyecto para el entorno de desarrollo debe llamarse *youruser-movies-dev*. El nuevo proyecto para el entorno de ensayo debe llamarse *youruser-movies-stage*.
- Las dependencias de Maven requeridas para compilar la aplicación están disponibles en:

`http://nexus-common.apps.cluster.domain.example.com/repository/java`

El archivo `settings.xml` en la raíz del repositorio Git de la aplicación contiene la dirección URL del servidor proxy Nexus que contiene todas las dependencias de Maven para compilar la aplicación. Debe editar este archivo y agregar la dirección URL del servidor Nexus.

- El microservicio implementado en el entorno de desarrollo debe estar disponible en la siguiente dirección URL:
`http://movies-youruser-movies-dev.apps.cluster.domain.example.com/movies`
- El microservicio implementado en el entorno de ensayo debe estar disponible en la siguiente dirección URL:
`http://movies-youruser-movies-stage.apps.cluster.domain.example.com/movies`

Pasos

Siga los pasos que se detallan a continuación para completar el trabajo de laboratorio:

1. Implemente una instancia efímera de Jenkins en el proyecto *youruser-jenkins*.
2. Configure el complemento de sincronización de Jenkins y permita que Jenkins monitoree el proyecto *youruser-movies-dev* en busca de tuberías.
3. Cree los proyectos de OpenShift para los entornos de desarrollo y ensayo.
4. Edite el archivo de recursos de configuración de compilación y agregue la dirección URL de Git del archivo de Jenkins.
5. Inspeccione la tubería de Jenkins. El archivo de Jenkins define una tubería de varias etapas para implementar el microservicio. Algunas de las etapas están completas, mientras que otras están incompletas.
Ingrase su clon local del repositorio Git D0288-apps y extraiga la bifurcación **maestra** del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido. Cree una nueva bifurcación denominada **review-cicd** para guardar los cambios que realice durante este ejercicio.

6. Revise y complete la tubería implementando las líneas marcadas como **TODO**: en el archivo de Jenkins. Edite el archivo `~/D0288-apps/movies/Jenkinsfile` y realice los siguientes pasos.
- 6.1. Abra el archivo `~/D0288-apps/movies/Jenkinsfile` con un editor de texto. También puede copiar las etapas completadas del archivo `~/D0288/solutions/review-cicd/Jenkinsfile` provisto.
 - 6.2. La tubería debe ejecutarse en un nodo de agente de Jenkins que admita Java y Apache Maven.
 - 6.3. Personalice la sección `environment` (entorno) y agregue valores según su entorno.
 - 6.4. Inspeccione la etapa `Launch new app in DEV env` (Iniciar aplicación nueva en entorno de desarrollo). Agregue código para iniciar una nueva aplicación de OpenShift en el entorno de desarrollo.
 - 6.5. Implemente la etapa `Wait for deployment in DEV env` (Esperar la implementación en el entorno de desarrollo). Agregue código para monitorear la implementación hasta que los pods estén en estado `Running` (En ejecución).
 - 6.6. Inspeccione la etapa `Promote to Staging Env` (Promover a entorno de ensayo). Agregue código para etiquetar la versión `latest` (más reciente) del flujo de imágenes del entorno de desarrollo con una etiqueta en el entorno de ensayo denominada `stage`.
 - 6.7. Inspeccione la etapa `Deploy to Staging Env` (Implementar en entorno de ensayo). Agregue código para crear una nueva aplicación en el entorno de ensayo con el flujo de imágenes etiquetado en la etapa anterior.
- El archivo de Jenkins ahora está completo. Guarde los cambios.
7. Edite el archivo de configuración del proxy de Maven en `~/D0288-apps/movies/settings.xml`, y cambie el valor de la etiqueta `url` para que apunte al servidor proxy de Nexus para su entorno.
8. Confirme los cambios al repositorio Git desde la carpeta `~/D0288-apps/movies`:

```
[student@workstation D0288-apps]$ cd ~/D0288-apps/movies
[student@workstation movies]$ git commit -a \
> -m "Completed Jenkinsfile for movies microservice"
...output omitted...
[student@workstation movies]$ git push
...output omitted...
[student@workstation movies]$ cd ~
```

9. Implemente la configuración de compilación en el proyecto `youruser-movies-dev` e inicie una nueva compilación. Asegúrese de agregar los roles de seguridad adecuados para los proyectos en los que Jenkins realiza alguna operación.
- Una vez que se ejecuta la compilación, puede monitorear el progreso de la tubería con la consola web de OpenShift. Se produce un error en la ejecución en la etapa `Run Unit Tests` (Ejecutar pruebas de unidad).

**nota**

Las tuberías de Jenkins quedaron obsoletas en OpenShift Container Platform 4. La funcionalidad equivalente está presente en las tuberías de OpenShift basadas en Tekton.

10. Investigue los errores de prueba mediante el registro de ejecución de tubería de Jenkins. Corrija las pruebas de unidad con errores y vuelva a ejecutar la tubería. La tubería ahora falla en la etapa **Static Code Analysis** (Análisis de código estático).
11. Corrija los errores señalados por las herramientas de análisis de código estático y, a continuación, vuelva a ejecutar la tubería.
12. Compruebe que el microservicio esté implementado y en ejecución en el entorno de desarrollo.
13. Promueva el microservicio al entorno de ensayo.
14. Compruebe que la ejecución de la tubería se realice correctamente. Compruebe que el microservicio se implemente y esté en ejecución en el entorno de ensayo.
15. Califique su trabajo.

Ejecute el siguiente comando en **workstation** para verificar que se hayan realizado todas las tareas:

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

16. Limpieza: Elimine los proyectos **youruser-jenkins**, **youruser-movies-dev** y **youruser-movies-stage**.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-movies-dev
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-movies-stage
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-jenkins
```

Finalizar

En **workstation**, ejecute el comando **lab todo-migrate finish** para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

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

Esto concluye el trabajo de laboratorio.

► Solución

Implementación de canalizaciones de integración continua y de implementación continua en OpenShift

Lista de verificación de rendimiento

En este trabajo de laboratorio, implementará Jenkins y creará una tubería de CI/CD para implementar un microservicio Java en OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implementar una instancia efímera de Jenkins en OpenShift.
- Crear un archivo de Jenkins con varias etapas para implementar el microservicio en el entorno de desarrollo.
- Promover el microservicio del entorno de desarrollo al entorno de ensayo.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación de muestra (movies) en el repositorio Git.
- La imagen del compilador S2I OpenJDK 8 requerida por la aplicación.

Ejecute el siguiente comando en la máquina virtual `workstation` para validar los requisitos previos y descargar los archivos de proyecto inicial y los archivos de soluciones:

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

Requisitos

El microservicio (movies) proporciona una API REST que enumera las películas. Está escrito en Java con el marco (framework) Spring Boot.

El microservicio se implementa de un código fuente almacenado en un repositorio Git.

Implementará una instancia efímera de Jenkins en OpenShift y la configurará para ejecutar tuberías desde espacios de nombres seleccionados en OpenShift.

Un archivo de recurso de configuración de compilación de inicio denominado `movies-bc.json` se ofrece en la carpeta `~/D0288/labs/review-cicd`. Debe editar este archivo y personalizarlo para su entorno.

En el repositorio Git se proporciona un archivo de Jenkins de inicio que contiene las etapas de tubería junto con el código fuente de la aplicación. Debe completar el archivo de Jenkins e implementarlo en OpenShift.

Una vez que implemente la tubería y la ejecute, verá errores en determinadas etapas. Corrija estos errores analizando el registro de ejecución de tubería de Jenkins y haga que la tubería ejecute todas las etapas correctamente.

Implemente el microservicio y la tubería de Jenkins en conformidad con los siguientes requisitos:

- Implemente una instancia efímera de Jenkins en un espacio de nombres denominado *youruser-jenkins*. Asegúrese de que el parámetro **MEMORY_LIMIT** de la plantilla está establecido en 2GB para mejorar el rendimiento.

- El nuevo proyecto para el entorno de desarrollo debe llamarse *youruser-movies-dev*. El nuevo proyecto para el entorno de ensayo debe llamarse *youruser-movies-stage*.

- Las dependencias de Maven requeridas para compilar la aplicación están disponibles en:

`http://nexus-common.apps.cluster.domain.example.com/repository/java`

El archivo `settings.xml` en la raíz del repositorio Git de la aplicación contiene la dirección URL del servidor proxy Nexus que contiene todas las dependencias de Maven para compilar la aplicación. Debe editar este archivo y agregar la dirección URL del servidor Nexus.

- El microservicio implementado en el entorno de desarrollo debe estar disponible en la siguiente dirección URL:

`http://movies-youruser-movies-dev.apps.cluster.domain.example.com/movies`

- El microservicio implementado en el entorno de ensayo debe estar disponible en la siguiente dirección URL:

`http://movies-youruser-movies-stage.apps.cluster.domain.example.com/movies`

Pasos

Siga los pasos que se detallan a continuación para completar el trabajo de laboratorio:

1. Implemente una instancia efímera de Jenkins en el proyecto *youruser-jenkins*.

- 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 OpenShift con su cuenta de usuario de desarrollador.

```
[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 proyecto nuevo para alojar la instancia de Jenkins. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-jenkins
Now using project "youruser-jenkins" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 1.4. Implemente una instancia de Jenkins.

```
[student@workstation ~]$ oc new-app --as-deployment-config \
> jenkins-ephemeral -p MEMORY_LIMIT=2048Mi
--> Deploying template "openshift/jenkins-ephemeral" to project youruser-jenkins
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
...output omitted...
```

- 1.5. La implementación llevará un tiempo. Compruebe que la instancia de Jenkins esté lista y en ejecución:

| NAME | READY | STATUS | RESTARTS | AGE |
|------------------|-------|----------------|----------|-------|
| jenkins-1-deploy | 0/1 | Completed | 0 | 2m42s |
| jenkins-1-rtnw5 | 1/1 | Running | 0 | 1m34s |

- 1.6. Obtenga la dirección URL de ruta para la instancia de Jenkins:

```
[student@workstation ~]$ oc get route/jenkins -o jsonpath='{.spec.host}{"\n"}'
jenkins-youruser-jenkins.apps.cluster.domain.example.com
```

- 1.7. Vaya a la dirección URL de ruta del paso anterior desde un navegador web.
Se le presentará una pantalla para iniciar sesión con sus credenciales de OpenShift. Haga clic en **Log in with OpenShift** (Iniciar sesión con OpenShift) para abrir la página de inicio de sesión de la consola web de OpenShift.
- 1.8. Inicie sesión con su cuenta de usuario de desarrollador. Su nombre de usuario es la variable RHT_OCP4_DEV_USER del archivo de configuración del aula /usr/local/etc/ocp4.config. Su contraseña es la variable RHT_OCP4_DEV_PASSWORD del mismo archivo.
- 1.9. Se mostrará una pantalla que le pedirá que autorice el acceso de la cuenta de servicio a su cuenta. Haga clic en **Allow selected permissions** (Permitir permisos seleccionados) para abrir la página de inicio de Jenkins.
2. Configure el complemento de sincronización de Jenkins y permita que Jenkins monitoree el proyecto youruser-movies-dev en busca de tuberías.
- 2.1. En la página de inicio de Jenkins, haga clic en **Manage Jenkins (Administrar Jenkins)** en el menú izquierdo de la página de inicio de Jenkins y, a continuación, haga clic en **Configure System (Configurar sistema)** en la página **Manage Jenkins (Administrar Jenkins)** para abrir la página de configuración global de Jenkins.

- 2.2. Desplácese hacia abajo hasta la sección **OpenShift Jenkins Sync** (Sincronización de OpenShift Jenkins) y agregue el espacio de nombres *youruser-movies-dev* al campo **Namespace** (Espacio de nombres), junto al espacio de nombres *youruser-jenkins* ya existente. Separe las entradas con un espacio y no use una coma.

No cambie otros parámetros en esta página. Creará el proyecto *youruser-movies-dev* en el paso siguiente.

Haga clic en **Save** (Guardar) para aplicar los cambios a la configuración global de Jenkins.

3. Cree los proyectos de OpenShift para los entornos de desarrollo y ensayo.
 - 3.1. Cree un nuevo proyecto para el entorno de ensayo. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-movies-stage
Now using project "youruser-movies-stage" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 3.2. Cree un nuevo proyecto para el entorno de desarrollo. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-movies-dev
Now using project "youruser-movies-dev" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

4. Edite el archivo de recursos de configuración de compilación y agregue la dirección URL de Git del archivo de Jenkins.

- 4.1. Inspeccione el archivo de recurso de configuración de compilación de tubería en `~/D0288/labs/review-cicd/movies-bc.json` con un editor de texto.

- 4.2. Edite la dirección URL de Git donde se almacena el código fuente de la aplicación.

Edite la configuración de compilación y cambie el valor del atributo `spec.source.git.uri` para que apunte al repositorio Git:

```
...output omitted...
git: {
  uri: "https://github.com/youruser/D0288-apps",
  ref: "review-cicd"
},
...output omitted...
```

5. Inspeccione la tubería de Jenkins. El archivo de Jenkins define una tubería de varias etapas para implementar el microservicio. Algunas de las etapas están completas, mientras que otras están incompletas.

Ingrese su clon local del repositorio Git `D0288-apps` y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido. Cree una nueva bifurcación denominada `review-cicd` para guardar los cambios que realice durante este ejercicio.

- 5.1. Observe la bifurcación maestra:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 5.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b review-cicd
Switched to a new branch 'review-cicd'
[student@workstation D0288-apps]$ git push -u origin review-cicd
...output omitted...
* [new branch]      review-cicd -> review-cicd
Branch review-cicd set up to track remote branch review-cicd from origin.
```

- 5.3. Inspeccione el archivo de Jenkins en ~/D0288-apps/movies/Jenkinsfile. El archivo de Jenkins define una tubería de varias etapas para implementar el microservicio. Algunas de las etapas están completas, mientras que otras están incompletas.

En los pasos siguientes, revise y complete la tubería implementando las líneas marcadas como **TODO**: en el archivo.

- 6.** Revise y complete la tubería implementando las líneas marcadas como **TODO**: en el archivo de Jenkins. Edite el archivo ~/D0288-apps/movies/Jenkinsfile y realice los siguientes pasos.

- 6.1. Abra el archivo ~/D0288-apps/movies/Jenkinsfile con un editor de texto. También puede copiar las etapas completadas del archivo ~/D0288/solutions/review-cicd/Jenkinsfile provisto.

- 6.2. La tubería debe ejecutarse en un nodo de agente de Jenkins que admite Java y Apache Maven.

Agregue una etiqueta a la sección node (nodo) para permitir que la tubería se ejecute en un nodo con la etiqueta maven. Localice la línea:

```
//TODO: Add label for the Maven jenkins agent
```

Reemplace el código con lo siguiente:

```
...output omitted...
agent {
  node {
    label 'maven'
  }
}
...output omitted...
```

- 6.3. Personalice la sección environment (entorno) y agregue valores según su entorno.

Reemplace el valor de youruser por su cuenta de usuario de desarrollador.

Reemplace el valor del atributo NEXUS_SERVER por el valor de la variable RHT_OCP4_NEXUS_SERVER del archivo /usr/local/etc/ocp4.config.

```
...output omitted...
environment {
    //TODO: Customize these variables for your environment
    DEV_PROJECT = "youruser-books-dev"
    STAGE_PROJECT = "youruser-books-stage"
    APP_GIT_URL = "https://github.com/youruser/D0288-apps"
    NEXUS_SERVER = "http://nexus-common.apps.cluster.domain.example.com/
repository/java"

    // DO NOT CHANGE THE GLOBAL VARS BELOW THIS LINE
    APP_NAME = "movies"
}
...output omitted...
```

- 6.4. Inspeccione la etapa Launch new app in DEV env (Iniciar aplicación nueva en entorno de desarrollo). Agregue código para iniciar una nueva aplicación de OpenShift en el entorno de desarrollo.

Localice las líneas:

```
// TODO: Create a new OpenShift application based on the ${APP_NAME}:latest image
stream
// TODO: Expose the ${APP_NAME} service for external access
```

Reemplácelas con las siguientes líneas:

```
...output omitted...
sh '''
  oc new-app --as-deployment-config ${APP_NAME}:latest -n ${DEV_PROJECT}
  oc expose svc/${APP_NAME} -n ${DEV_PROJECT}
'''
...output omitted...
```

- 6.5. Implemente la etapa Wait for deployment in DEV env (Esperar la implementación en el entorno de desarrollo). Agregue código para monitorear la implementación hasta que los pods estén en estado Running (En ejecución).

Localice la línea:

```
//TODO: Watch deployment until pod is in 'Running' state
```

Reemplácela con las siguientes líneas:

```

...output omitted...
steps {
  script {
    openshift.withCluster() {
      openshift.withProject( "${DEV_PROJECT}" ) {
        openshift.selector("dc", "${APP_NAME}").related('pods').untilEach(1) {
          return (it.object().status.phase == "Running")
        }
      }
    }
  }
...output omitted...

```

- 6.6. Inspeccione la etapa Promote to Staging Env (Promover a entorno de ensayo). Agregue código para etiquetar la versión latest (más reciente) del flujo de imágenes del entorno de desarrollo con una etiqueta en el entorno de ensayo denominada stage.

Localice la línea:

```
// TODO: Tag the ${APP_NAME}:latest image stream in the dev env as
${APP_NAME}:stage in staging
```

Reemplácela con las siguientes líneas:

```

...output omitted...
openshift.tag("${DEV_PROJECT}/${APP_NAME}:latest", "${STAGE_PROJECT}/
${APP_NAME}:stage")
...output omitted...

```

- 6.7. Inspeccione la etapa Deploy to Staging Env (Implementar en entorno de ensayo). Agregue código para crear una nueva aplicación en el entorno de ensayo con el flujo de imágenes etiquetado en la etapa anterior.

Localice la línea:

```
// TODO: Create a new app in staging and expose the service
```

Reemplácela con las siguientes líneas:

```

...output omitted...
sh '''
  oc project ${STAGE_PROJECT}
  oc new-app --as-deployment-config --name ${APP_NAME} -i ${APP_NAME}:stage
  oc expose svc/${APP_NAME}
'''
...output omitted...

```

El archivo de Jenkins ahora está completo. Guarde los cambios.

7. Edite el archivo de configuración del proxy de Maven en ~/D0288-apps/movies/settings.xml, y cambie el valor de la etiqueta url para que apunte al servidor proxy de Nexus para su entorno.

Localice la etiqueta <url> debajo de la línea:

```
<!-- // TODO: Change the url attribute to the nexus proxy server URL for your
environment. -->
```

Agregue el valor de la variable RHT_OCP4_NEXUS_SERVER del archivo /usr/local/etc/ocp4.config a la etiqueta <url>:

```
...output omitted...
<url>http://nexus-common.apps.cluster.domain.example.com/repository/java</url>
...output omitted...
```

8. Confirme los cambios al repositorio Git desde la carpeta ~/D0288-apps/movies:

```
[student@workstation D0288-apps]$ cd ~/D0288-apps/movies
[student@workstation movies]$ git commit -a \
> -m "Completed Jenkinsfile for movies microservice"
...output omitted...
[student@workstation movies]$ git push
...output omitted...
[student@workstation movies]$ cd ~
```

9. Implemente la configuración de compilación en el proyecto youruser-movies-dev e inicie una nueva compilación. Asegúrese de agregar los roles de seguridad adecuados para los proyectos en los que Jenkins realiza alguna operación.

Una vez que se ejecuta la compilación, puede monitorear el progreso de la tubería con la consola web de OpenShift. Se produce un error en la ejecución en la etapa Run Unit Tests (Ejecutar pruebas de unidad).



nota

Las tuberías de Jenkins quedaron obsoletas en OpenShift Container Platform 4. La funcionalidad equivalente está presente en las tuberías de OpenShift basadas en Tekton.

- 9.1. Asegúrese de crear la configuración de compilación en el espacio de nombres youruser-movies-dev.

```
[student@workstation ~]$ oc project ${RHT_OCP4_DEV_USER}-movies-dev
Now using project "youruser-movies-dev" on server
...output omitted...
```

- 9.2. Agregue el rol edit (edición) a la cuenta de servicio asociada con la implementación de Jenkins para los proyectos youruser-movies-dev y youruser-movies-stage.

```
[student@workstation ~]$ oc policy add-role-to-user \
> edit system:serviceaccount:${RHT_OCP4_DEV_USER}-jenkins:jenkins \
> -n ${RHT_OCP4_DEV_USER}-movies-dev
clusterrole.rbac.authorization.k8s.io/edit added:
"system:serviceaccount:youruser-jenkins:jenkins"

[student@workstation ~]$ oc policy add-role-to-user \
```

```
> edit system:serviceaccount:${RHT_OCP4_DEV_USER}-jenkins:jenkins \
> -n ${RHT_OCP4_DEV_USER}-movies-stage
clusterrole.rbac.authorization.k8s.io/edit added:
"system:serviceaccount:youruser-jenkins:jenkins"
```

- 9.3. Cree la configuración de la compilación:

```
[student@workstation ~]$ oc create \
> -f ~/DO288/labs/review-cicd/movies-bc.json
buildconfig.build.openshift.io/movies-pipeline created
```

- 9.4. Abra un navegador web y navegue en la consola web de OpenShift.

Inspeccione las rutas en el proyecto `openshift-console` para encontrar el nombre de host de su consola web de OpenShift.

```
[student@workstation ~]$ oc get route console -n openshift-console \
> -o jsonpath='{.spec.host}{"\n"}'
console-openshift-console.apps.cluster.domain.example.com
```

Inicie sesión con su cuenta de usuario de desarrollador.

- 9.5. Seleccione el proyecto `youruser-movies-dev`.

En la barra de navegación izquierda, haga clic en **Builds (Compilaciones)** → **Build Configs (Configuraciones de compilación)** y compruebe que una configuración de compilación denominada `movies-pipeline` esté visible.

- 9.6. Inicie una nueva compilación desde la línea de comandos para iniciar la tubería:

```
[student@workstation ~]$ oc start-build movies-pipeline
build.build.openshift.io/movies-pipeline-1 started
```

También puede usar la consola web de OpenShift para iniciar una compilación. Haga clic en la configuración de compilación `movies-pipeline` y, a continuación, haga clic en **Actions (Acciones)** → **Start Build (Iniciar compilación)** en la esquina superior derecha para iniciar una compilación.

- 10.** Investigue los errores de prueba mediante el registro de ejecución de tubería de Jenkins. Corrija las pruebas de unidad con errores y vuelva a ejecutar la tubería. La tubería ahora falla en la etapa **Static Code Analysis** (Análisis de código estático).
- 10.1. En el menú izquierdo de la consola web de OpenShift, haga clic en **Builds** → **Builds** para abrir la página **Builds (Compilaciones)** de la aplicación. Haga clic en `movies-pipeline-1` para abrir la página **Build Details** (Detalles de compilación). En la página **Build Details** (Detalles de compilación), haga clic en **View Logs** (Ver registros, debajo de **Build 1** [Compilación 1] con la marca roja 'x') para ver y abrir el registro de ejecución de tubería de Jenkins.
- 10.2. En el registro de ejecución de tubería de Jenkins, se muestra un error en la etapa **Run Unit Tests (Ejecutar pruebas de unidad)**:

```

...output omitted...
[ERROR] Tests run: 4, Failures: 2, Errors: 0, Skipped: 0, Time elapsed: 10.401 s
<<< FAILURE! - in com.redhat.movies.MoviesApplicationTests
[ERROR] testGetAllMovies(com.redhat.movies.MoviesApplicationTests)  Time elapsed:
0.172 s  <<< FAILURE!
java.lang.AssertionError: expected:<7> but was:<6>
at com.redhat.movies.MoviesApplicationTests.testGetAllMovies
(MoviesApplicationTests.java:52)

[ERROR] testGetStatus(com.redhat.movies.MoviesApplicationTests)  Time elapsed:
0.008 s  <<< FAILURE!
org.junit.ComparisonFailure: expected:<[Ready]> but was:<[OK]>
at com.redhat.movies.MoviesApplicationTests.testGetStatus
(MoviesApplicationTests.java:65)
...output omitted...

```

Debería ver dos pruebas fallando. Los métodos de prueba con errores, junto con los números de línea en el código fuente, se proporcionan en el registro de errores. El primer error indica que hay seis películas en el catálogo, pero la prueba afirma erróneamente que hay siete. El segundo error indica que la respuesta de la API es la cadena **OK** (Aceptar), pero la prueba afirma que la respuesta es **Ready** (Listo).

- 10.3. Corrija las pruebas de unidad que fallan. Abra el archivo `~/D0288-apps/movies/src/test/java/com/redhat/movies/MoviesApplicationTests.java` en un editor de texto.
- 10.4. Busque la línea siguiente en el método de prueba `testGetAllMovies()`:

```
Assert.assertEquals(7, movies.size());
```

Reemplace el código con la siguiente afirmación correcta:

```
Assert.assertEquals(6, movies.size());
```

- 10.5. Busque la línea siguiente en el método de prueba `testGetStatus()`:

```
Assert.assertEquals("Ready", response.getBody());
```

Reemplace el código con la siguiente afirmación correcta:

```
Assert.assertEquals("OK", response.getBody());
```

Guarde los cambios.

- 10.6. Confirme los cambios y envíe el archivo de prueba fijo a la bifurcación `youruser-review-cicd`.

```

[student@workstation ~]$ cd ~/D0288-apps/movies
[student@workstation movies]$ git commit -a -m "Fixed failed unit test."
...output omitted...
[student@workstation movies]$ git push
...output omitted...
[student@workstation movies]$ cd ~

```

- 10.7. Inicie una nueva compilación para volver a ejecutar la tubería.

```
[student@workstation ~]$ oc start-build movies-pipeline
build.build.openshift.io/movies-pipeline-2 started
```

- 10.8. Monitoree el progreso de la ejecución de la tubería para la compilación **movies-pipeline-2** desde la página **Build Details** (Detalles de compilación).

Las pruebas de unidad ahora deberían aprobarse. La tubería ahora falla en la etapa **Static Code Analysis** (Análisis de código estático).

- 11.** Corrija los errores señalados por las herramientas de análisis de código estático y, a continuación, vuelva a ejecutar la tubería.

- 11.1. Compruebe el registro de ejecución de tubería de Jenkins con un clic en **View Logs** (Ver registros). Los registros muestran que la herramienta de análisis de código estático se queja de las importaciones y variables no usadas en el código.

```
...output omitted...
[INFO] --- maven-pmd-plugin:3.12.0:check (default-cli) @ movies ---
[INFO] PMD Failure: com.redhat.movies.MoviesController:5 Rule:UnusedImports
Priority:4 Avoid unused imports such as 'java.io.File'.
[INFO] PMD Failure: com.redhat.movies.MoviesController:17 Rule:UnusedPrivateField
Priority:3 Avoid unused private fields such as 'flag'..
...output omitted...
```

- 11.2. Corrija el código de la clase identificada por la herramienta de análisis de código estático. Abra el archivo `~/D0288-apps/movies/src/main/java/com/redhat/movies/MoviesController.java` en un editor de texto.

- 11.3. Elimine la siguiente línea de la sección de importaciones en la parte superior del archivo:

```
import java.io.File;
```

- 11.4. Elimine la siguiente línea del archivo:

```
private String flag = "READY";
```

Guarde los cambios.

- 11.5. Confirme los cambios y envíe el archivo de prueba fijo a la bifurcación `youruser-review-cicd`.

```
[student@workstation ~]$ cd ~/D0288-apps/movies
[student@workstation movies]$ git commit -a \
> -m "Removed unused imports and variables."
...output omitted...
[student@workstation movies]$ git push
...output omitted...
[student@workstation movies]$ cd ~
```

- 11.6. Inicie una nueva compilación para volver a ejecutar la tubería.

```
[student@workstation ~]$ oc start-build movies-pipeline
build.build.openshift.io/movies-pipeline-3 started
```

- 11.7. Monitoree el progreso de la ejecución de la tubería para la compilación **movies-pipeline-3** desde la página **Build Details** (Detalles de compilación).

La tubería pasa la etapa de análisis de código estático y espera la aprobación en la etapa **Promote to Staging Env** (Promover a entorno de ensayo).

12. Compruebe que el microservicio esté implementado y en ejecución en el entorno de desarrollo.
- 12.1. Antes de promover el microservicio al entorno de ensayo, compruebe que la aplicación se esté ejecutando en el entorno de desarrollo.

Verifique que el microservicio **movies** se esté ejecutando:

```
[student@workstation ~]$ oc get pods -n ${RHT_OCP4_DEV_USER}-movies-dev
NAME           READY   STATUS    RESTARTS   AGE
...output omitted...
movies-1-jpjvc  1/1     Running   0          17m
```

- 12.2. Compruebe que el entorno de ensayo no tenga ningún pod en ejecución:

```
[student@workstation ~]$ oc get pods -n ${RHT_OCP4_DEV_USER}-movies-stage
No resources found.
```

- 12.3. Obtenga la dirección URL de ruta del microservicio **movies**:

```
[student@workstation ~]$ oc get route/movies \
> -n ${RHT_OCP4_DEV_USER}-movies-dev \
> -o jsonpath='{.spec.host}{"\n"}'
movies-youruser-movies-dev.apps.cluster.domain.example.com
```

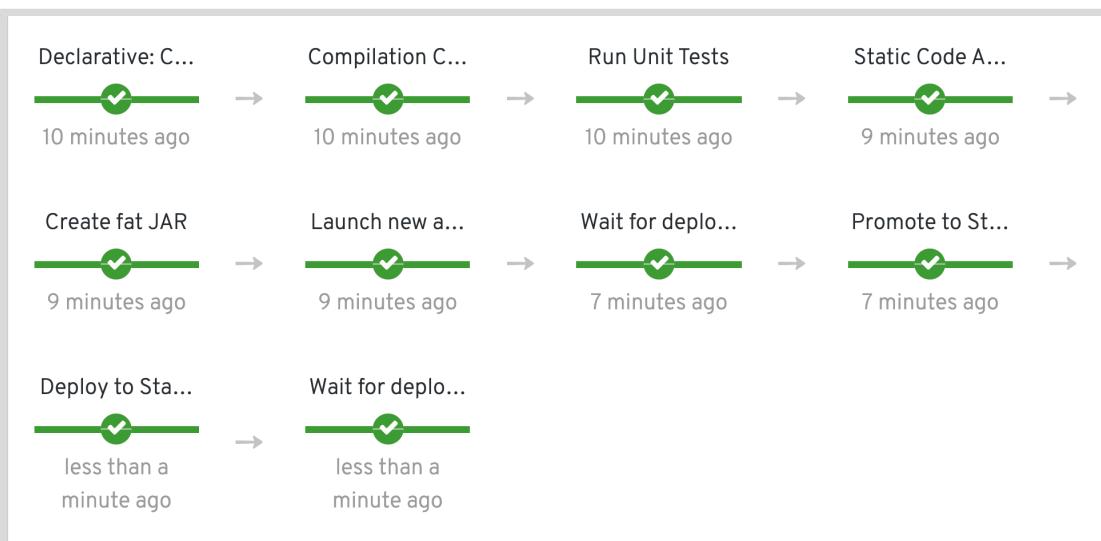
- 12.4. Vaya al extremo (endpoint) **/movies** en la dirección URL de ruta del paso anterior y compruebe que el microservicio **movies** muestra datos de películas.

```
[student@workstation ~]$ curl \
> movies-${RHT_OCP4_DEV_USER}-movies-dev.${RHT_OCP4_WILDCARD_DOMAIN}/movies
[ {
  "movieId" : 1,
  "name" : "The Godfather",
  "genre" : "Crime/Thriller"
}, {
  "movieId" : 2,
  "name" : "Star Wars",
  "genre" : "Sci-Fi"
...output omitted...
  "movieId" : 6,
  "name" : "The Silence of the Lambs",
  "genre" : "Drama"
} ]
```

13. Promueva el microservicio al entorno de ensayo.
- 13.1. Una vez que esté satisfecho de que el microservicio funciona según lo previsto, promueva el microservicio al entorno de ensayo.
 - 13.2. En la página **Build Details** (Detalles de compilación) de la consola web de OpenShift para la compilación **movies-pipeline-3**, haga clic en **Input Required** (Entrada requerida) para ver el registro de ejecución de tubería de Jenkins.
 - 13.3. En el menú izquierdo de la consola de Jenkins, haga clic en **Paused for Input** (Pausado para entrada). Haga clic en **Proceed** (Continuar) en el prompt **Move to Staging?** (¿Pasar a ensayo?).
 - 13.4. Jenkins cambia a la ventana de registro de ejecución y continúa ejecutando la etapa **Deploy to Staging Env** (Implementar en entorno de ensayo).
14. Compruebe que la ejecución de la tubería se realice correctamente. Compruebe que el microservicio se implemente y esté en ejecución en el entorno de ensayo.
- 14.1. La ejecución de la tubería debe completarse y debería ver lo siguiente en el registro de ejecución:

```
...output omitted...
Deployment to Staging env is complete.
Access the API endpoint at the URL
http://movies-youruser-movies-stage.apps.cluster.domain.example.com/movies.
...output omitted...
[Pipeline] End of Pipeline
Finished: SUCCESS
```

- 14.2. Haga clic en la dirección URL del microservicio movies en el entorno de ensayo. Compruebe que el microservicio sea similar en funciones a la versión implementada en el entorno de desarrollo.
- 14.3. En la página **Build Details** (Detalles de compilación) para **movies-pipeline-3**, se muestra el estado completo de ejecución de la tubería.



15. Califique su trabajo.

Ejecute el siguiente comando en `workstation` para verificar que se hayan realizado todas las tareas:

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

- 16.** Limpieza: Elimine los proyectos `youruser-jenkins`, `youruser-movies-dev` y `youruser-movies-stage`.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-movies-dev  
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-movies-stage  
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-jenkins
```

Finalizar

En `workstation`, ejecute el comando `lab todo-migrate finish` para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

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

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Una tubería de CI/CD es un proceso automatizado para integrar e implementar código con frecuencia en producción en un proyecto con varios desarrolladores.
- OpenShift Container Platform tiene compatibilidad integrada para implementar tuberías de CI/CD mediante Jenkins.
- Las tuberías constan de varias etapas y puede controlar dónde se ejecuta cada etapa mediante agentes.
- El complemento (plug-in) de cliente de Jenkins de OpenShift proporciona un DSL fácil de usar para crear scripts de tuberías e interactuar con un clúster OpenShift.

Compilación de aplicaciones para OpenShift

Meta

Crear e implementar aplicaciones en OpenShift.

Objetivos

- Integrar una aplicación contenedorizadas con servicios no contenedorizados.
- Implementar aplicaciones de terceros contenedorizadas siguiendo las prácticas recomendadas para OpenShift.
- Usar un tiempo de ejecución de aplicación de Red Hat OpenShift para implementar una aplicación.

Secciones

- Integración de servicios externos (y ejercicio guiado)
- Implementación de aplicaciones contenedorizadas (y ejercicio guiado)
- Implementación de aplicaciones con Red Hat OpenShift Application Runtimes (y ejercicio guiado)

Trabajo de laboratorio

Compilación de aplicaciones nativas de la nube para OpenShift

Integración de servicios externos

Objetivos

Tras finalizar esta sección, deberá ser capaz de integrar una aplicación contenedizada con servicios no contenedorizados.

Revisión de los Servicios Red Hat OpenShift

Un típico servicio en OpenShift tiene el nombre y un selector. Un servicio usa su selector para identificar pods que deben recibir solicitudes de aplicaciones enviadas al servicio. Las aplicaciones OpenShift usan el nombre del servicio para conectarse a los extremos (endpoints) de servicio.

De forma similar, un FQDN permite que una aplicación use un nombre para tener acceso a los extremos (endpoints) de un servicio público. Sin embargo, los servicios OpenShift permiten el acceso de la aplicación a los extremos (endpoints) de servicio sin requerir la exposición pública del servicio.

Una aplicación descubre un servicio usando variables de entorno o el servidor DNS interno OpenShift. El uso de variables de entorno requiere que el servicio se defina antes de la creación del pod de la aplicación; de lo contrario, la aplicación no recibirá las variables de entorno.

El uso del servidor DNS interno de OpenShift es más flexible porque permite que las aplicaciones detecten servicios dinámicamente. El nombre de servicio se vuelve un nombre de host DNS para todos los pods dentro del mismo clúster OpenShift que contiene el servicio. OpenShift agrega el sufijo del dominio `svc.cluster.local` a la ruta de búsqueda del sistema de resolución de DNS de todos los contenedores. OpenShift también asigna el nombre de host `service-name.project-name.svc.cluster.local` a cada servicio.

Por ejemplo, si el servicio `myapi` existe en el servicio `myproject`, todos los pods en el mismo clúster OpenShift pueden resolver el nombre del host `myapi.myproject.svc.cluster.local` para obtener la dirección IP del servicio. También se cuenta con los siguientes nombres cortos de host:

- Los pods del mismo proyecto pueden usar el nombre de servicio `myapi` como nombre corto de host, sin ningún sufijo de dominio.
- Los pods de un proyecto diferente pueden usar el nombre de servicio y el nombre de proyecto `myapi.myproject` como nombre corto de host, sin el sufijo de dominio `svc.cluster.local`.

Definición de servicios externos

OpenShift admite varios enfoques para definir servicios que no contienen selectores. Esto permite que un servicio OpenShift indique uno o más hosts fuera del clúster OpenShift.

Considere una aplicación que desea contener, que depende de un servicio de base de datos existente que no está disponible fácilmente dentro del clúster OpenShift. No es necesario migrar el servicio de base de datos a OpenShift antes de contenedorizar la aplicación. En su lugar, comience a diseñar la aplicación para interactuar con los servicios de OpenShift, incluido el servicio de base de datos. Simplemente, cree un servicio OpenShift que haga referencia a los extremos (endpoints) del servicio de base de datos externa.

Al crear servicios OpenShift para los extremos (endpoints) de servicios externos, las aplicaciones pueden detectar servicios internos y externos. Además, si los extremos (endpoints) de un servicio externo cambian, no es necesario volver a configurar las aplicaciones afectadas. En su lugar, actualice los extremos (endpoints) para el servicio OpenShift correspondiente.

Creación de un servicio externo

El enfoque más sencillo para crear un servicio externo es usar el comando `oc create service externalname` con la opción `--external-name`:

```
[user@host ~]$ oc create service externalname myservice \
> --external-name myhost.example.com
```

El ejemplo anterior también puede aceptar una dirección IP en lugar de un nombre DNS.

A continuación, las aplicaciones que se ejecutan dentro del clúster OpenShift usan el nombre de servicio de la misma forma que usarían un servicio regular, ya sea como una variable de entorno o como un nombre de host local.

Definición de extremos (endpoints) para un servicio

Un servicio típico crea recursos de extremo (endpoint) en forma dinámica, en función del atributo del selector del servicio. Los comandos `oc status` y `oc get all` no muestran estos recursos. Puede usar el comando `oc get endpoints` para visualizarlos.

Si usa el comando `oc create service externalname --external-name` para crear un servicio, el comando también crea un recurso de extremo (endpoint) que apunta al nombre del host o la dirección IP dada como argumento.

Si no usa la opción `--external-name`, no crea un recurso de extremo (endpoint). En este caso, use el comando `oc create -f` y un archivo de definición de recursos para crear de manera explícita los recursos de extremo (endpoint).

Si crea un extremo (endpoint) desde un archivo, puede definir varias direcciones IP para el mismo servicio externo y usar las funciones de balanceo de carga del servicio de OpenShift. En este escenario, OpenShift no agrega ni elimina direcciones para dar cuenta de la disponibilidad de cada instancia. Una aplicación externa debe actualizar la lista de direcciones IP en el recurso de extremo (endpoint).

Para obtener más información acerca de los archivos de definición de recursos de extremo (endpoint), consulte las referencias al final de esta sección.



Referencias

Revise la sección *Type ExternalName* de la documentación de conceptos del Servicio para Kubernetes en
<https://kubernetes.io/docs/concepts/services-networking/service/#externalname>

Revise las convenciones de nomenclatura DNS de OpenShift en el capítulo *Redes* de la *Guía de arquitectura* para Red Hat OpenShift Container Platform 4.5 en
https://access.redhat.com/documentation/en-us/openshift_container_platform/4.5/html/networking/index

Nota de los autores: *El siguiente recurso aún no existe en la versión 4.x de la documentación de OpenShift. Sin embargo, la información enlazada a continuación sigue siendo relevante para la versión 4.5 actual del producto.*

Consulte el capítulo *Integración de servicios externos* de la *Guía para desarrolladores* para Red Hat OpenShift Container Platform 3.11 en
https://docs.openshift.com/container-platform/3.11/dev_guide/integrating_external_services.html

► Ejercicio Guiado

Integración de un servicio externo

En este ejercicio, implementará una aplicación en OpenShift que se comunica con la base de datos que se ejecuta fuera del clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Implemente la aplicación To Do List del código fuente.
- Cree un servicio de base de datos para la aplicación que apunta al servidor de la base de datos MariaDB fuera del clúster OpenShift.
- Verifique que la aplicación use los datos precargados en la base de datos.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación To Do List en el repositorio Git (`todo-single`).
- Un servidor Nexus que proporciona las dependencias npm requeridas por la aplicación (`restify`, `sequelize` y `mysql`).
- Imagen de compilador S2I Node.js 12.
- Un servidor de base de datos MariaDB relleno previamente que se ejecuta fuera del clúster OpenShift.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos e implemente la aplicación To Do List:

```
[student@workstation ~]$ lab external-service start
```

El comando anterior ejecuta el script `oc-new-app.sh` en la carpeta `~/D0288/labs/external-service` para implementar la aplicación. Puede revisar este script si necesita más información acerca de los recursos de la aplicación To Do List que usa durante este ejercicio.

► 1. Inspeccione los recursos de aplicaciones To Do List.

1.1. Cargue la configuración de su entorno de aula.

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

1.2. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

13. Escriba el proyecto `youruser-external-service` que hospeda la aplicación To Do List:

```
[student@workstation ~]$ oc project ${RHT_OCP4_DEV_USER}-external-service
Now using project "youruser-external-service" on server
"https://api.cluster.domain.example.com:6443".
```

14. Verifique que el proyecto OpenShift cuente con una única aplicación denominada `todoapp`, la cual está compilada a partir de fuentes:

```
[student@workstation ~]$ oc status
...output omitted...
http://todo-youruser-external-service.apps.domain.cluster.example.com to pod port
8080-tcp (svc/todoapp)
dc/todoapp deploys istag/todoapp:latest <-
bc/todoapp source builds https://github.com/youruser/D0288-apps on openshift/
nodejs:12
deployment #1 deployed 26 seconds ago - 1 pod
...output omitted...
```

15. Verifique que la aplicación esté lista y en ejecución:

```
[student@workstation ~]$ oc get pod
NAME          READY   STATUS    RESTARTS   AGE
todoapp-1-6z6qg  1/1     Running   0          1m
todoapp-1-build  0/1     Completed  0          4m
todoapp-1-deploy 0/1     Completed  0          1m
```

16. Inspeccione las variables del entorno dentro del pod de la aplicación para obtener los parámetros de conexión de la base de datos:

```
[student@workstation ~]$ oc rsh todoapp-1-6z6qg env | grep DATABASE
DATABASE_PASSWORD=redhat123
DATABASE_SVC=tododb
DATABASE_USER=todoapp
DATABASE_INIT=false
DATABASE_NAME=todo
```

- 2. Compruebe que la aplicación To Do list no puede llegar al servidor de base de datos especificado por la variable de entorno `DATABASE_SVC`, que es `tododb`.

- 2.1. Obtenga el nombre de host donde se expone la aplicación. Dado que el nombre del host es muy largo, guárdelo en una variable de shell.

```
[student@workstation ~]$ HOSTNAME=$(oc get route todoapp \
> -o jsonpath='{.spec.host}')
[student@workstation ~]$ echo ${HOSTNAME}
todoapp-youruser-external-service.apps.cluster.domain.example.com
```

- 2.2. Use el comando `curl`, el nombre de host del paso anterior y la ruta de acceso de recursos de API `/todo/api/items/6` para capturar un ítem de la base de datos. El mensaje de error de la aplicación indica que no puede resolver el nombre de host del servicio `tododb`.

```
[student@workstation ~]$ curl -si http://${HOSTNAME}/todo/api/items/6
HTTP/1.1 500 Internal Server Error
...output omitted...
{"message":"getaddrinfo ENOTFOUND tododb tododb:3306"}
```

► 3. Inspeccione la base de datos externa.

- 3.1. Busque el nombre de host del servidor MariaDB externo. Este nombre de host es el mismo que el dominio comodín del clúster de OpenShift, y reemplaza el prefijo `apps.` con `mysql.ocp-`.

```
[student@workstation ~]$ dbhost=$(echo \
> mysql.ocp-$RHT_OCP4_WILDCARD_DOMAIN#"apps.")
[student@workstation ~]$ echo ${dbhost}
mysql.ocp-cluster.domain.example.com
```

- 3.2. Use el comando `mysqlshow` para conectarse a la base de datos `todo` en el servidor MariaDB externo y compruebe que contiene la tabla `Item` (ítem).

Use el nombre del host del paso anterior. Use el usuario `todoapp` y `redhat123` como contraseña:

```
[student@workstation ~]$ mysqlshow -h${dbhost} \
> -utodoapp -predhat123 todo
Database: todo
+-----+
| Tables |
+-----+
| Item   |
+-----+
```

► 4. Cree un servicio OpenShift que se conecte a la instancia de la base de datos externa y verifique que la aplicación ahora obtenga los datos de la base de datos externa.

- 4.1. Use el comando `oc create svc` para crear un servicio basado en un nombre externo y el nombre de host del servidor de base de datos del paso anterior.

```
[student@workstation ~]$ oc create svc externalname tododb \
> --external-name ${dbhost}
service/tododb created
```

- 4.2. Verifique que el servicio tododb exista y muestre una IP externa, pero ninguna IP de clúster:

```
[student@workstation ~]$ oc get svc
NAME      TYPE        CLUSTER-IP      EXTERNAL-IP
todoapp   ClusterIP   172.30.140.201  <none>
tododb    ExternalName <none>        mysql.ocp-cluster.domain.example.com ...
```

- 4.3. Verifique que la aplicación ahora pueda obtener datos de la base de datos externa.

Use el comando curl de nuevo:

```
[student@workstation ~]$ curl -si http://${HOSTNAME}/todo/api/items/6
HTTP/1.1 200 OK
...
{"id":6,"description":"Verify that the To Do List application works","done":false}
```

- 5. Realice la limpieza. Elimine el proyecto *youruser-external-service* de OpenShift.

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-external-service
```

Finalizar

En workstation, ejecute el comando `lab external-service finish` para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab external-service finish
```

Esto concluye el ejercicio guiado.

Servicios de contenedорización

Objetivos

Después de completar esta sección, los estudiantes deberán ser capaces de revisar e implementar aplicaciones de terceros contenedorizadas siguiendo las prácticas recomendadas para OpenShift.

Revisión de aplicaciones contenedorizadas existentes en OpenShift

Ocasionalmente, debe implementar una aplicación en OpenShift cuyo proveedor proporcione un Dockerfile para compilar su imagen de contenedor o proporcione una imagen de contenedor precompilada en un servidor de registro. Esta aplicación puede ofrecer un servicio backend que requiere aplicación personalizada, como una base de datos o un sistema de mensajería, o un servicio que requiere el proceso de desarrollo, como un registro de contenedor hospedado, un repositorio de artefactos o una herramienta de colaboración.

El hecho de que la aplicación ya esté contenedorizada por su proveedor no significa necesariamente que se ejecuta bien en OpenShift. El primer intento de implementar esa aplicación mediante el comando `oc new-app` puede producir un error por una de varias de las razones posibles. La resolución de estos problemas puede requerir la personalización de la configuración de implementación de la aplicación o cambios en su Dockerfile.

La siguiente es una lista no exhaustiva de problemas comunes:

- La aplicación no se ejecuta bajo las políticas de seguridad openShift predeterminadas definidas por la restricción del contexto de seguridad (SCC) `restricted`. La imagen de contenedor de la aplicación puede esperar ejecutarse como usuario `root` o cualquier otro usuario fijo. También puede requerir políticas SELinux personalizadas y otros ajustes con privilegios.

Normalmente, puede cambiar el Dockerfile para que la aplicación cumpla con las políticas de seguridad predeterminadas de OpenShift. Si no es así, evalúe cuidadosamente el riesgo de ejecutar la aplicación bajo una política de seguridad más relajada.

- La aplicación requiere una configuración personalizada para las solicitudes de recursos y los límites de recursos. Esto suele ocurrir con aplicaciones monolíticas heredadas que no se diseñaron para una arquitectura basada en microservicios. Los administradores de clústeres de OpenShift suelen configurar intervalos de límite predeterminados que proporcionan valores predeterminados para las solicitudes de recursos y los límites del proyecto, y estos pueden ser demasiado pequeños para la aplicación heredada.

Puede especificar el aumento de las solicitudes de recursos y los límites de recursos en la configuración de implementación para invalidar los valores predeterminados establecidos por el administrador del clúster. Si los requisitos de la aplicación superan la cuota de recursos establecida por los administradores del clúster OpenShift, los administradores deben aumentar la cuota de recursos para el usuario o proporcionar una cuenta de servicio para la aplicación con una cuota de recursos aumentada.

- La imagen de la aplicación define valores de configuración que están bien para un tiempo de ejecución de contenedor independiente, pero no para un openshift o un clúster de Kubernetes genérico. Las aplicaciones suelen leer la configuración de las variables de entorno y, por lo general, definen los valores predeterminados mediante instrucciones ENV en un Dockerfile .

No es necesario cambiar el Dockerfile para cambiar sus variables de entorno. Puede invalidar cualquier variable de entorno, incluidas las que no se establecen explícitamente mediante instrucciones ENV en el Dockerfile, en la configuración de implementación de la aplicación.

- La aplicación requiere almacenamiento persistente. Las aplicaciones suelen definir los requisitos de almacenamiento mediante instrucciones VOLUME en Dockerfile. A veces, estas aplicaciones esperan usar carpetas de host para sus volúmenes, lo que normalmente está prohibido por los administradores del clúster OpenShift. Estas aplicaciones pueden esperar tener acceso directo al almacenamiento de red, que tampoco se recomienda desde un punto de vista de seguridad.

La manera correcta de proporcionar almacenamiento persistente para las aplicaciones en OpenShift es definir los recursos de la reclamación de volumen persistente (PVC) y configurar la configuración de implementación de la aplicación para asociar estos PVC a los volúmenes del contenedor de aplicaciones. OpenShift administra el almacenamiento de red en nombre de las aplicaciones y los administradores de clúster administran las políticas de seguridad y los niveles de rendimiento para el almacenamiento.

El proveedor puede proporcionar archivos de recursos de Kubernetes para implementar aplicaciones, y estos archivos de recursos pueden requerir personalización para funcionar en OpenShift. Es posible que los recursos de implementación que funcionan con Kubernetes upstream (y también con otras distribuciones de proveedores de Kubernetes) no funcionen con OpenShift porque Kubernetes estándar no viene con la configuración predeterminada segura.

A veces, los proveedores proporcionan recursos de implementación de Kubernetes suponiendo que el usuario es un administrador de clúster. Estos proveedores pueden no considerar que un clúster OpenShift normalmente es compartido por varias organizaciones y equipos, a los que no se le concederían privilegios de administrador de clúster para su propia seguridad.

Revisión del Dockerfile para la aplicación Nexus

La aplicación Nexus, desarrollada por Sonatype, es un administrador de repositorios generalmente usado por desarrolladores para almacenar artefactos requeridos por las aplicaciones, como dependencias. Esta aplicación puede almacenar artefactos para varias tecnologías como Java, .NET, Docker, Node, Python y Ruby.

Sonatype ofrece un Dockerfile para compilar la aplicación Nexus usando una imagen de contenedor de base Red Hat Linux. La imagen resultante está disponible en Red Hat Container Catalog, lo que atestigua que cumple con un conjunto básico (core) de directrices de Red Hat. El Dockerfile está configurado para admitir las funciones de OpenShift, como permitir que las imágenes de contenedor se ejecuten como cuenta de usuario aleatoria.

Metadata de imagen base y contenedor

El proyecto Dockerfile está disponible en <https://github.com/sonatype/docker-nexus3/tree/3.18.0>. En el archivo Dockerfile.rh.el, se proporciona lo siguiente:

```
...output omitted...
FROM      registry.access.redhat.com/rhel7/rhel ①

MAINTAINER Sonatype <cloud-ops@sonatype.com>
...output omitted...

LABEL name="Nexus Repository Manager" \
...output omitted...
io.k8s.description="The Nexus Repository Manager server \
```

```

        with universal support for popular component formats." \❷
io.k8s.display-name="Nexus Repository Manager" \
io.openshift.expose-services="8081:8081" \
io.openshift.tags="Sonatype,Nexus,Repository Manager"

...output omitted...

```

- ❶ Especifica la imagen de contenedor de Red Hat Linux 7 como la imagen base. También hay un archivo Dockerfile alternativo que usa Red Hat Universal Base Image 8 ubicado en el proyecto GitHub.
- ❷ Define los metadatos de imágenes para Kubernetes y OpenShift.

Nexus Dockerfile también usa la instrucción Dockerfile ARG. Una instrucción ARG define una variable que solo existe durante el proceso de compilación de imágenes. A diferencia de una instrucción ENV, estas variables no forman parte de la imagen de contenedor resultante:

```

...output omitted...
ARG NEXUS_VERSION=3.18.0-01
ARG NEXUS_DOWNLOAD_URL=https://.../nexus/3/nexus-${NEXUS_VERSION}-unix.tar.gz
ARG NEXUS_DOWNLOAD_SHA256_HASH=e1d9...c99e
...output omitted...

```

Los procesos de compilación usan estas tres variables para controlar y verificar la versión instalada de Nexus en la imagen del contenedor.

Dockerfile también define las variables de entorno que están presentes en la imagen de contenedor compilada:

```

...output omitted...
# configure nexus runtime
ENV SONATYPE_DIR=/opt/sonatype
ENV NEXUS_HOME=${SONATYPE_DIR}/nexus \
    NEXUS_DATA=/nexus-data \
    NEXUS_CONTEXT='' \
    SONATYPE_WORK=${SONATYPE_DIR}/sonatype-work \
DOCKER_TYPE='rh-docker'
...output omitted...

```

El proceso de instalación usa la variable de entorno DOCKER_TYPE para personalizar la instalación y la configuración. Un valor de rh-docker invoca los cambios de configuración necesarios para una imagen de contenedor certificada por Red Hat.

El proceso de instalación de Nexus

Sonatype mantiene libros de recetas de Chef que estandarizan la instalación de la aplicación Nexus. Chef es una tecnología de gestión de configuración de código abierto, similar a Puppet y Ansible, que tiene como objetivo agilizar el proceso de configuración y gestión de servidores. El *recetario* de Chef es una colección de *recetas* de Chef, y cada receta define la configuración de un conjunto determinado de recursos del sistema.

Nexus Dockerfile usa el proceso de instalación de Chef para compilar la imagen del contenedor Nexus:

```

...output omitted...
ARG NEXUS_REPOSITORY_MANAGER_COOKBOOK_VERSION="release-0.5.20190212-..."1
ARG NEXUS_REPOSITORY_MANAGER_COOKBOOK_URL="https://github.com/sonatype/..."

ADD solo.json.erb /var/chef/solo.json.erb2

# Install using chef-solo
RUN curl -L https://www.getchef.com/chef/install.sh | bash \3
&& /opt/chef/embedded/bin/erb /var/chef/solo.json.erb > /var/chef/solo.json \
&& chef-solo \
--node_name nexus_repository_red_hat_docker_build \
--recipe-url ${NEXUS_REPOSITORY_MANAGER_COOKBOOK_URL} \
--json-attributes /var/chef/solo.json \
&& rpm -qa *chef* | xargs rpm -e \
&& rpm --rebuilddb \
&& rm -rf /etc/chef \
&& rm -rf /opt/chefdk \
&& rm -rf /var/cache/yum \
&& rm -rf /var/chef
...output omitted...

```

- 1** El Dockerfile usa variables para controlar la versión del recetario de Chef que instala Nexus. La dirección URL del recetario se proporciona como argumento para un comando de Chef en una instrucción RUN posterior.
- 2** En el mismo directorio que el Dockerfile, el archivo `solo.json.erb` contiene detalles de configuración de Chef. El Dockerfile agrega este archivo a la imagen de contenedor para habilitar el proceso de instalación de Chef.
- 3** La instalación de Nexus se realiza con una sola instrucción RUN que:
 - Descarga e instala Chef.
 - Ejecuta el recetario de Chef para instalar Nexus con el comando `chef -solo`.
 - Elimina los archivos descargados, los RPM de Chef y otros archivos extraños.

El recetario de Chef también configura los permisos de archivo para que los datos y las carpetas de registro se puedan escribir mediante `gui=0`, cumpliendo así con las políticas de seguridad predeterminadas de OpenShift.

Entorno de ejecución del contenedor

La parte inferior de Dockerfile proporciona metadatos para permitir que un tiempo de ejecución de contenedor cree un contenedor a partir de la imagen:

```

...output omitted...
VOLUME ${NEXUS_DATA}1

EXPOSE 80812
USER nexus3

ENV INSTALL4J_ADD_VM_PARAMS="-Xms1200m -Xmx1200m ..."4

ENTRYPOINT ["/uid_entrypoint.sh"]5
CMD ["sh", "-c", "${SONATYPE_DIR}/start-nexus-repository-manager.sh"]6

```

- ➊ Configura el directorio `/nexus-data` como un volumen, porque el valor de la variable `NEXUS_DATA` es `/nexus-data`. La aplicación Nexus almacena los datos y recursos de la aplicación en este directorio.
- ➋ La aplicación del nexo comunica sobre el puerto 8081.
- ➌ La aplicación Nexus se ejecuta como el usuario del nexo. El proceso de instalación de Chef crea y configura este usuario.
- ➍ La aplicación Nexus funciona en una máquina virtual Java (JVM). La JVM necesita opciones para dimensionar su montón cuando se ejecuta en un tiempo de ejecución de contenedor independiente. En OpenShift, debe invalidar estas opciones y dejar que la JVM obedezca los límites de recursos y las solicitudes de recursos de OpenShift. Si no lo hace, puede resultar en problemas de programación y estabilidad.
- ➎ Durante la instalación, el recetario de Chef crea el script `/uid_entrypoint.sh`. El objetivo del script `/uid_entrypoint.sh` es reconocer el ID de usuario que está ejecutando la aplicación y definirlo para el usuario `nexus`. Esto permite que la aplicación se ejecute normalmente con la restricción de contexto de seguridad `restringida` predeterminada.
- ➏ El comando que inicia la aplicación Nexus. El punto de entrada ejecuta este comando después de ajustar los permisos para tener en cuenta el UID de proceso asignado aleatoriamente.

Personalización de recursos OpenShift para la imagen Nexus

Si crea una imagen de contenedor Nexus a partir de su Dockerfile oficial y la implementa mediante un comando como `oc new-app --as-deployment-config --docker-image`, es posible que no funcione de forma fiable. Para asegurarse de que la imagen de contenedor funcione bien, debe realizar algunas personalizaciones en su configuración de implementación.

Establecer solicitudes y límites de recursos

Red Hat recomienda que las aplicaciones implementadas en OpenShift definan solicitudes y límites de recursos. Si no está usando versiones muy antiguas de la máquina virtual Java (JVM), no es necesario definir opciones de montón de JVM.

Las solicitudes de recursos muestran la cantidad de CPU y memoria que necesita ejecutar una aplicación. El planificador de OpenShift busca un nodo trabajador en el clúster con suficiente CPU y memoria disponibles para ejecutar la aplicación, por lo que no se podrá iniciar con un error de memoria insuficiente.

Los límites de recursos indican cuánto puede aumentar el uso de CPU y memoria de una aplicación con el tiempo. OpenShift establece cgroups del kernel de Linux para el contenedor de aplicaciones, lo que evita que infrinja esos límites. El kernel de Linux elimina los contenedores que infringen estos límites y OpenShift inicia los contenedores de reemplazo, mitigando así los problemas causados por fugas de memoria, bucles infinitos e interbloqueos. Las versiones recientes de la JVM dimensionan automáticamente el montón y otras estructuras de datos internas para no infringir la configuración actual de cgroups.

Las solicitudes de recursos y los límites de recursos se definen dentro de la plantilla de pod de una configuración de implementación:

```
...output omitted...
- apiVersion: v1
  kind: DeploymentConfig
...output omitted...
  spec:
...output omitted...
```

```

template:
...output omitted...
spec:
  containers:
...output omitted...
  resources:
    limits:
      cpu: "1"
      memory: 2Gi
    requests:
      cpu: 500m
      memory: 256Mi
...output omitted...

```

En el escenario del nexo, usted debe también invalidar la variable de entorno `INSTALL4J_ADD_VM_PARAMS` para quitar cualquier opción JVM relacionada con el tamaño de la memoria:

```

...output omitted...
- apiVersion: v1
  kind: DeploymentConfig
...output omitted...
  spec:
...output omitted...
  template:
...output omitted...
  spec:
    containers:
      - env:
          - name: INSTALL4J_ADD_VM_PARAMS
            value: -Djava.util.prefs.userRoot=/nexus-data/javaprefs
...output omitted...

```

En el ejemplo anterior, se supone que no está reemplazando la variable de entorno `NEXUS_DATA` para especificar una ruta de acceso de volumen diferente para el almacenamiento persistente del contenedor.

Implementación de sondas

Red Hat recomienda que las aplicaciones implementadas en OpenShift definan sondeos de estado. Nexus ofrece una API para determinar si el servicio está activo y si cuenta con interbloqueos. Sin embargo, no puede usar un sondeo HTTP simple, porque la API de Nexus requiere autenticación y la API devuelve un código de estado HTTP 200 incluso si Nexus está en mal estado.

Puede crear un script que se autentique con la API de Nexus y analice las respuestas de la API. El siguiente script verifica si el servicio Nexus está listo para atender solicitudes:

```

#!/bin/sh
curl -si -u admin:$(cat /nexus-data/admin.password) \ ①
  http://localhost:8081/service/metrics/ping | grep pong ②

```

① La opción `-u` pasa un nombre de usuario y una contraseña en la solicitud HTTP GET.

- ❷ Nexus ofrece el extremo (endpoint) `/service/metrics/ping` que debe devolver el valor `pong` cuando el servicio está listo. Si `pong` no está en la respuesta, el comando `grep` se cierra con un código de estado distinto de cero.

El usuario `admin` está autorizado para realizar solicitudes a los extremos (endpoints) de comprobación de estado. Cuando la aplicación Nexus se inicializa por primera vez, genera una contraseña aleatoria para el usuario `admin`. Nexus almacena la contraseña aleatoria en el archivo `/nexus-data/admin.password`.

De manera similar, el script siguiente determina si el servicio Nexus está activo:

```
#!/bin/sh
curl -si -u admin:$(cat /nexus-data/admin.password) \
    http://localhost:8081/service/metrics/healthcheck | grep healthy | \
    grep true
```

Debido a que estos scripts de sondeo son simples, puede colocar el contenido de cada script en la estrofa de sondeo adecuada de la especificación del contenedor Nexus:

```
...output omitted...
- apiVersion: v1
  kind: DeploymentConfig
...output omitted...
  spec:
...output omitted...
  template:
...output omitted...
  spec:
    containers:
...output omitted...
    livenessProbe:
      exec:
        command:
          - /bin/sh ①
          - "-c" ②
          - > ③
            curl -si -u admin:$(cat /nexus-data/admin.password)
            http://localhost:8081/service/metrics/healthcheck |
            grep healthy | grep true
      failureThreshold: 3
      initialDelaySeconds: 10
      periodSeconds: 10
...output omitted...
```

- ❶ Usar la shell `/bin/sh`.
- ❷ La opción `-c` indica que se ejecuta un único comando en la shell.
- ❸ Indicador de continuación multilínea YAML. Esto le permite dividir una cadena larga en varias líneas.

Para un script de sondeo grande, considere la posibilidad de modificar el Dockerfile y agregar los scripts de sondeo a la imagen de contenedor.



Referencias

Dispone de información adicional acerca del servicio Nexus en la Documentación del administrador de repositorios Nexus en
<https://help.sonatype.com/repomanager3>

Para obtener más información sobre el estado y las métricas de las aplicaciones Nexus, consulte

Información sobre el estado y las métricas del servidor Nexus 3 – Soporte de Sonatype

<https://support.sonatype.com/hc/en-us/articles/226254487-Nexus-3-Server-Metrics-and-Health-Information>

Para obtener más información sobre la configuración de la memoria JVM en contenedores, consulte

Java dentro de docker: lo que debe saber para no FALLAR

<https://developers.redhat.com/blog/2017/03/14/java-inside-docker/>

► Ejercicio Guiado

Implementación de un servidor Nexus contenedorizado

En este ejercicio, implementará un servidor Nexus como aplicación contenedorizada siguiendo las prácticas recomendadas por OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Compile una imagen de contenedores desde un Dockerfile.
- Verifique que las plantillas usen un flujo de imágenes que use una imagen de contenedor de un registro privado.
- Compruebe que una plantilla define las solicitudes y los límites de recursos para una aplicación Java y reemplaza las variables de entorno que establecerían tamaños de montón.
- Configure una plantilla para usar un reclamo de volumen persistente.
- Configure una plantilla para usar los sondeos de ejecución y disponibilidad.
- Inicie una nueva aplicación usando el archivo de la plantilla.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Red Hat Universal Base Image 8 (ubi8/ubi).
- Los archivos de origen para la imagen de la aplicación Nexus en el repositorio Git (nexus3).

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab nexus-service start
```

► 1. Revise y compile el archivo Dockerfile de Nexus.

- 1.1. Ingrese el subdirectorio `nexus3` de su clon local del repositorio Git D0288-apps y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd ~/D0288-apps/nexus3
[student@workstation nexus3]$ git checkout master
...output omitted...
```

- 1.2. Inspeccione el archivo `Dockerfile` y observe que la aplicación Nexus mantiene sus archivos en la carpeta `/nexus-data`:

```
[student@workstation nexus3]$ grep VOLUME Dockerfile
VOLUME ${NEXUS_DATA}
[student@workstation nexus3]$ grep NEXUS_DATA Dockerfile
  NEXUS_DATA=/nexus-data \
...output omitted...
```

El Dockerfile define la variable de entorno `NEXUS_DATA` con un valor de `/nexus-data`. El Dockerfile usa esta variable para definir un volumen donde Nexus almacena los datos de la aplicación.

- 1.3. Inspeccione el archivo `Dockerfile` y observe que defina una variable de entorno que establezca los tamaños de montón de las máquinas virtuales Java (JVM).
Más adelante, debe invalidar esta variable de entorno para que la configuración de implementación de OpenShift controle la configuración de memoria del pod.

```
[student@workstation nexus3]$ grep ENV Dockerfile
...output omitted...
ENV INSTALL4J_ADD_VM_PARAMS="-Xms1200m -Xmx1200m -XX:MaxDirectMemorySize=2g -
Djava.util.prefs.userRoot=${NEXUS_DATA}/javaprefs"
```

- 1.4. 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 nexus3]$ source /usr/local/etc/ocp4.config
```

- 1.5. Cree la imagen del contenedor y llévela a su cuenta personal en Quay.io. Puede ejecutar o copiar y pegar los comandos desde el script `build-nexus-image.sh` hasta el script `~/D0288/labs/nexus-service`.

Durante la compilación, ignore la advertencia "HOSTNAME is not supported for OCI image format" (Nombre de host no soportado para el formato de imagen OCI).

```
[student@workstation nexus3]$ sudo podman build -t nexus3 .
...output omitted...
STEP 33: COMMIT nexus3
[student@workstation nexus3]$ sudo podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
[student@workstation nexus3]$ sudo skopeo copy \
> containers-storage:localhost/nexus3 \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/nexus3
...output omitted...
Writing manifest to image destination
Storing signatures
```

**Importante**

El proceso de compilación no está pensado para tardar más de 5 minutos en completarse. Durante las pruebas de este ejercicio, se notificaron tiempos de compilación mucho más largos.

Si la compilación no se completa después de cinco minutos, escriba **Ctrl+C** para salir del proceso de compilación. Use el siguiente comando para copiar una imagen de Nexus precompilada directamente en su cuenta de Quay.io personal y omitir el proceso de compilación:

```
[student@workstation nexus3]$ sudo skopeo copy \
> docker://quay.io/redhattraining/nexus3:latest \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/nexus3
```

- ▶ 2. Complete la plantilla para implementar Nexus como servicio.

La plantilla parcial se ofrece para implementar Nexus. Actualice la plantilla para implementar Nexus como servicio.

- 2.1. Cree una copia de la plantilla de inicio para editar, y salga de la carpeta `~/D0288-apps/nexus3`.

```
[student@workstation nexus3]$ cp ~/D0288/labs/nexus-service/nexus-template.yaml \
> ~/nexus-template.yaml
[student@workstation nexus3]$ cd ~
[student@workstation ~]$
```

- 2.2. Inspeccione el archivo `~/nexus-template.yaml` para comprobar que usa la imagen de contenedor compilada anteriormente.

```
[student@workstation ~]$ grep -A1 "kind: DockerImage" ~/nexus-template.yaml
      kind: DockerImage
      name: quay.io/youruser/nexus3:latest
```

- 2.3. Inspeccione el archivo `~/nexus-template.yaml` para comprobar que define la solicitud y los límites de recursos para el pod de aplicación:

```
[student@workstation ~]$ grep -B1 -A5 limits: ~/nexus-template.yaml
  resources:
    limits:
      cpu: "1"
      memory: 2Gi
    requests:
      cpu: 500m
      memory: 256Mi
```

- 2.4. Abra el archivo `~/nexus-template.yaml` con un editor de texto y, a continuación, reemplace la variable de entorno `INSTALL4J_ADD_VM_PARAMS` para no incluir ninguna configuración de tamaño de montón de JVM y para definir solo las propiedades del sistema Java requeridas por la aplicación.

```

...output omitted...
- apiVersion: v1
kind: DeploymentConfig
...output omitted...
- env:
  - name: INSTALL4J_ADD_VM_PARAMS
    value: -Djava.util.prefs.userRoot=/nexus-data/javaprefs
...output omitted...

```

- 2.5. La aplicación Nexus contiene un extremo (endpoint) /service/metrics/healthcheck que comprueba que la aplicación esté activa. El acceso a un extremo (endpoint) requiere autenticación. Cuando se inicia la aplicación, la contraseña de usuario admin se almacena en el archivo /nexus-data/admin.password.

Revise el sondeo de ejecución en el recurso de configuración de implementación. Abra el archivo ~/nexus-template.yaml con un editor de texto y configure el sondeo para iniciarse 120 segundos después de iniciado el contenedor. Además, configure el sondeo para un tiempo máximo de espera de ejecución de 30 segundos.

```

...output omitted...
- apiVersion: v1
kind: DeploymentConfig
...output omitted...
  livenessProbe:
    exec:
      command:
        - /bin/sh
        - "-c"
        - '>
          curl -siu admin:$(cat /nexus-data/admin.password)
          http://localhost:8081/service/metrics/healthcheck |
          grep healthy | grep true
        ...output omitted...
    initialDelaySeconds: 120
    ...output omitted...
    timeoutSeconds: 30
...output omitted...

```

- 2.6. La aplicación Nexus contiene un extremo (endpoint) /service/metrics/ping que comprueba que la aplicación esté lista. El acceso a este extremo (endpoint) además requiere autenticación.

Revise el sondeo de disponibilidad en el recurso de configuración de implementación. Abra el archivo ~/nexus-template.yaml con un editor de texto y configure el sondeo para iniciarse 120 segundos después de iniciado el contenedor. Además, configure el sondeo para un tiempo máximo de espera de ejecución de 30 segundos.

```

...output omitted...
- apiVersion: v1
kind: DeploymentConfig
...output omitted...
  readinessProbe:
    exec:
      command:

```

```

- /bin/sh
- "-c"
- >
curl -siu admin:$(cat /nexus-data/admin.password)
http://localhost:8081/service/metrics/ping |
grep pong
...output omitted...
initialDelaySeconds: 120
...output omitted...
timeoutSeconds: 30
...output omitted...

```

- 2.7. En el recurso de configuración de implementación, configure el montaje del volumen para usar el volumen **nexus-data** en la ruta del montaje **/nexus-data**:

```

...output omitted...
- apiVersion: v1
kind: DeploymentConfig
...output omitted...
volumeMounts:
- mountPath: /nexus-data
  name: nexus-data
...output omitted...

```

- 2.8. En el recurso de configuración de implementación, nombre el volumen **nexus-data**; luego, configure el volumen para usar la reclamación de volumen persistente **nexus-data-pvc**:

```

...output omitted...
- apiVersion: v1
kind: DeploymentConfig
...output omitted...
volumes:
- name: nexus-data
  persistentVolumeClaim:
    claimName: nexus-data-pvc
...output omitted...

```

- 2.9. Se requiere una reclamación de volumen persistente para continuar con los datos de Nexus. En el recurso de reclamación de volumen persistente, nombre la reclamación de volumen persistente **nexus-data-pvc** actualizando el atributo **metadata.name**:

```

...output omitted...
- apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  labels:
    app: ${NEXUS_SERVICE_NAME}
  name: nexus-data-pvc
...output omitted...

```

- 2.10. El archivo de la plantilla ahora está completo. Guarde sus ediciones.

Para comprobar los cambios realizados durante este paso, compare el archivo de plantilla con el archivo de plantilla de solución en `~/D0288/solutions/nexus-service/nexus-template.yaml`. Si no está seguro acerca de sus ediciones, puede copiar el archivo de soluciones y continuar con el siguiente paso.

- 3. Cree un proyecto nuevo. Agregue un secreto que permita a cualquier usuario del proyecto extraer la imagen del contenedor del nexo de Quay.io.

- 3.1. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 3.2. Cree un nuevo proyecto para la aplicación:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-nexus-service
Now using project "yourname-nexus-service" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 3.3. Use Podman sin el comando sudo para iniciar sesión en Quay.io.

```
[student@workstation ~]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password:
Login Succeeded!
```

- 3.4. Cree un secreto para acceder a su cuenta personal de Quay.io.

```
[student@workstation ~]$ oc create secret generic quayio \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
secret/quayio created
```

- 3.5. Vincule el secreto con la cuenta de servicio default (predeterminada).

```
[student@workstation ~]$ oc secrets link default quayio --for pull
```

- 4. Cree una nueva aplicación.

- 4.1. Cree una nueva aplicación denominada `nexus3` desde el archivo de plantilla. Pase como parámetro `HOSTNAME` un valor que se base en el nombre de usuario del desarrollador y el dominio comodín del clúster OpenShift.
Puede copiar el comando completo o ejecutarlo directamente desde `/home/student/D0288/labs/nexus-service/oc-new-app.sh`.

```
[student@workstation ~]$ oc new-app --as-deployment-config --name nexus3 \
> -f ~/nexus-template.yaml \
> -p HOSTNAME=nexus-${RHT_OCP4_DEV_USER}.${RHT_OCP4_WILDCARD_DOMAIN}
```

```
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "nexus3" created
deploymentconfig.apps.openshift.io "nexus3" created
service "nexus3" created
route.route.openshift.io "nexus3" created
persistentvolumeclaim "nexus-data-pvc" created
--> Success
...output omitted...
```

- 4.2. Espere hasta que el pod de aplicación esté en ejecución, pero no esté listo. Siga los registros del pod para observar el largo procedimiento de inicialización del servidor Nexus. Cuando vea el mensaje "Iniciado", puede dejar de seguir el registro.

```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
nexus3-1-deploy  0/1     Completed   0          4m34s
nexus3-1-kfwwh   0/1     Running    0          1m25s
[student@workstation ~]$ oc logs -f nexus3-1-kfwwh
...output omitted...
-----
Started Sonatype Nexus OSS 3.18.0-01
-----
...output omitted...
```

Presione Ctrl+C para salir del comando `oc logs`.

- 4.3. Espere a que la aplicación esté lista y en ejecución. OpenShift puede tardar unos segundos en obtener un estado correcto de los sondeos de estado de la aplicación.

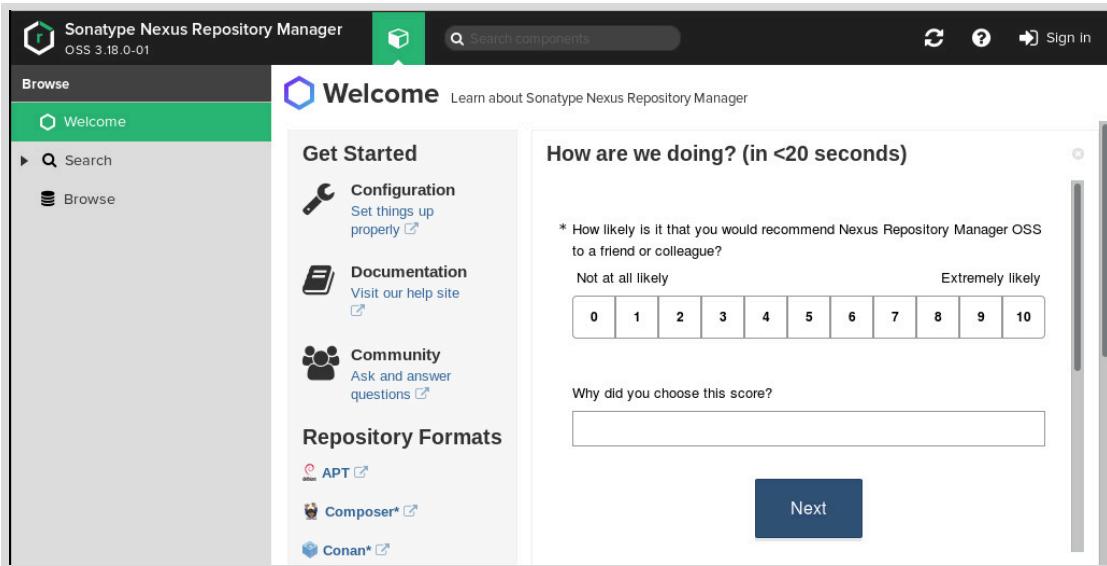
```
[student@workstation ~]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
nexus3-1-deploy  0/1     Completed   0          4m34s
nexus3-1-kfwwh   1/1     Running    0          4m25s
```

► 5. Pruebe la aplicación Nexus.

- 5.1. Recupere la ruta para la aplicación Nexus:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT
nexus3    nexus-yourname.apps.cluster.domain.example.com ...
```

- 5.2. Abra un navegador web y navegue a la ruta de aplicación. En la página, se muestra la aplicación Nexus.



Si desea iniciar sesión como usuario `admin`, debe obtener la contraseña del archivo `/nexus-data/admin.password` en el contenedor. No es necesario iniciar sesión como parte de esta prueba.

- 6. Elimine el proyecto en OpenShift así como el contenedor y la imagen en el registro de contenedores externo. Dado que Quay.io permite recuperar imágenes de contenedor antiguas, también debe eliminar el repositorio en Quay.io.

6.1. Elimine el proyecto de OpenShift:

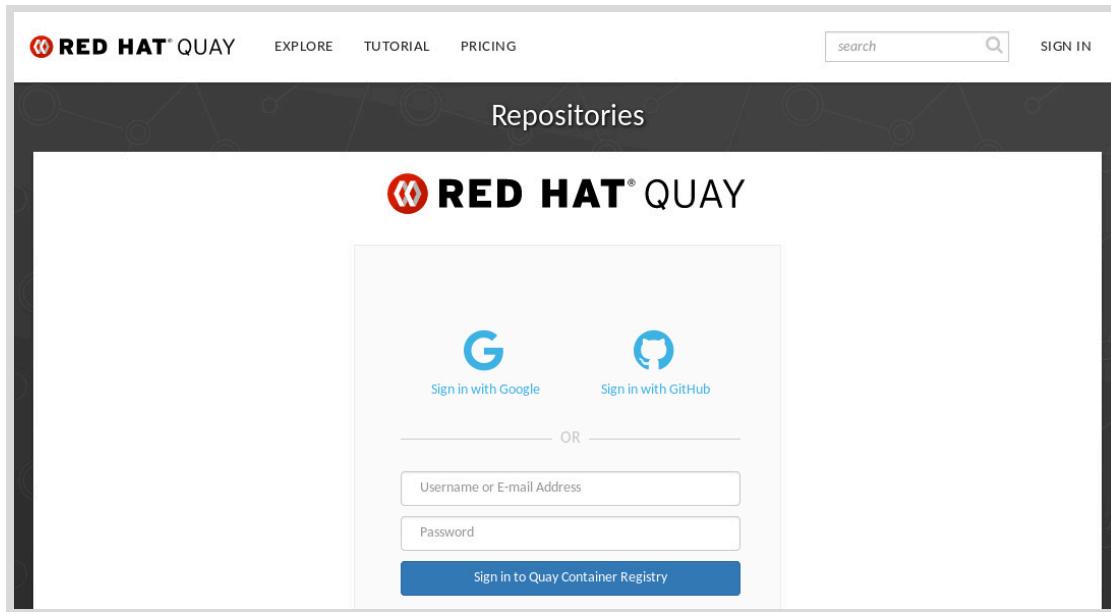
```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-nexus-service
project.project.openshift.io "youruser.nexus-service" deleted
```

6.2. Elimine la imagen de contenedor del registro externo:

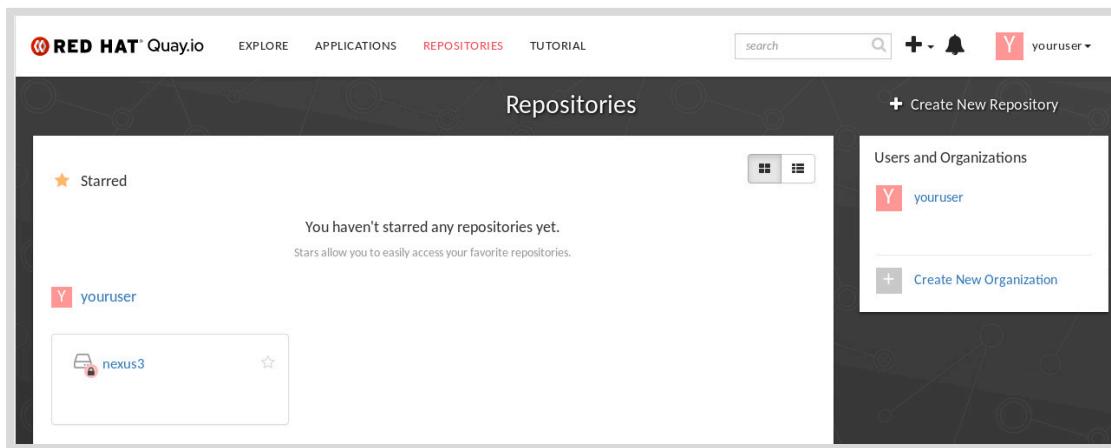
```
[student@workstation ~]$ skopeo delete \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/nexus3
```

6.3. Inicie sesión en Quay.io con su cuenta personal gratuita.

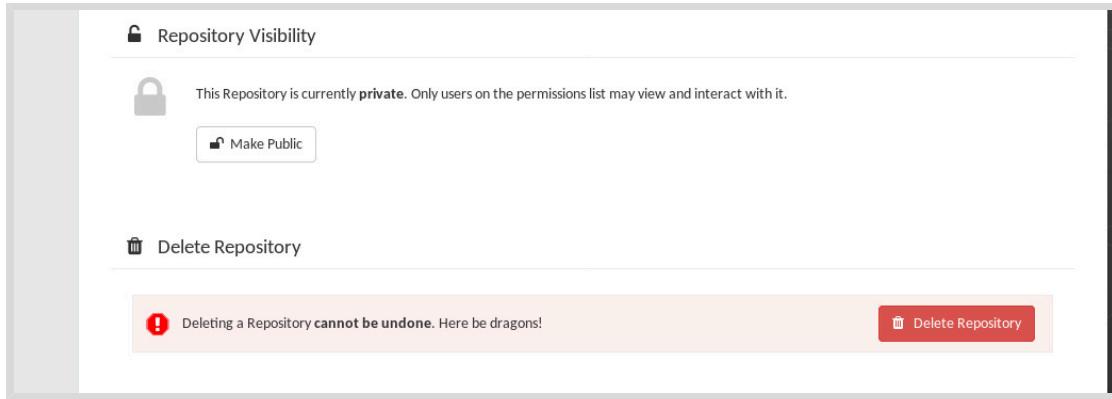
Navegue hasta <https://quay.io>; luego, haga clic en **Sign In** (Iniciar sesión) para proporcionar sus credenciales de usuario. Haga clic en **Sign in to Quay Container Registry** (Iniciar sesión en el registro de contenedor de Quay) para iniciar sesión en Quay.io.



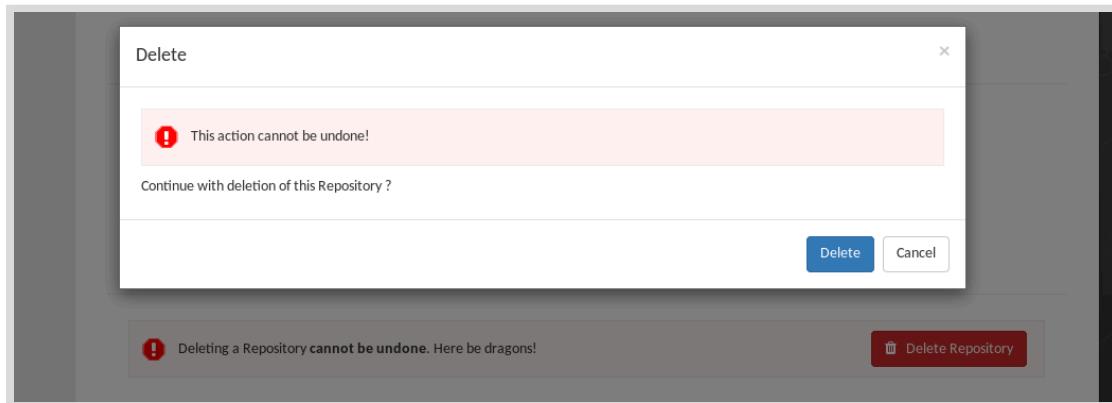
- 6.4. En el menú principal de Quay.io, haga clic en **Repositories** (Repositorios) y busque **nexus**. El ícono de candado indica que es un repositorio privado que requiere autenticación para extracciones e inserciones. Haga clic en **nexus3** para mostrar la página **Repository Activity** (Actividad del repositorio).



- 6.5. En la página **Repository Activity** (Actividad del repositorio) del repositorio **nexus3**, desplácese hacia abajo y haga clic en el ícono de engranaje para mostrar la pestaña **Settings** (Configuración). En la pestaña **Settings** (Configuración), desplácese hacia abajo y haga clic en **Delete Repository** (Eliminar repositorio).



- 6.6. En el cuadro de diálogo Delete (Eliminar), haga clic en Delete para confirmar que desea eliminar el repositorio **nexus3**. Después de unos momentos, volverá a la página **Repositories** (Repositorios). Cierre sesión en Quay.io.



Finalizar

En **workstation**, ejecute el comando **lab nexus-service finish** para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab nexus-service finish
```

Esto concluye el ejercicio guiado.

Implementación de aplicaciones con Red Hat OpenShift Application Runtimes

Objetivos

Después de completar esta sección, deberá ser capaz de implementar una aplicación mediante un tiempo de ejecución de aplicación de Red Hat OpenShift.

Desarrollo de aplicaciones nativas en la nube

Cloud Native Computing Foundation (CNCF) define la nube nativa como:

Las tecnologías nativas de la nube permiten a las organizaciones crear y ejecutar aplicaciones escalables en entornos modernos y dinámicos, como nubes públicas, privadas e híbridas. Los contenedores, las mallas de servicio, los microservicios, la infraestructura inmutable y las API declarativas ejemplifican este enfoque.

Estas técnicas permiten sistemas acoplados libremente que son resilientes, manejables y observables. En combinación con una automatización sólida, permiten a los ingenieros realizar cambios de alto impacto con frecuencia y predicción con un mínimo trabajo.

El CNCF identifica a Kubernetes como una tecnología nativa de la nube y estableció un estándar de certificación para distribuciones de Kubernetes. Cuando desarrolla e implementa aplicaciones en una distribución de Kubernetes certificada, evita el bloqueo de la plataforma porque las cargas de trabajo son portátiles y funcionan con otras distribuciones certificadas. Para mantener la certificación, las distribuciones deben garantizar actualizaciones oportunas para mantener la paridad de características con el proyecto Kubernetes de código abierto.

OpenShift Container Platform es una distribución Kubernetes certificada por CNCF. Una aplicación que se implementa en OpenShift y que está diseñada para usar los servicios proporcionados por la plataforma se considera una aplicación nativa de la nube.

Red Hat OpenShift Application Runtimes

Red Hat OpenShift Application Runtimes (RHOAR) es una recopilación de bases de tiempo de ejecución compiladas previamente y contenedezorizadas para compilar aplicaciones nativas en la nube en Red Hat OpenShift. RHOAR proporciona un enfoque optimizado y soportado con Red Hat para desarrollar aplicaciones de microservicios destinadas a OpenShift como plataforma de implementación.

RHOAR admite varios tiempos de ejecución, lenguajes, marcos (frameworks) y arquitecturas. Ofrece las opciones y la flexibilidad para elegir los marcos (frameworks) y tiempos de ejecución adecuados para el trabajo adecuado. Las aplicaciones desarrolladas con RHOAR pueden ejecutarse en cualquier infraestructura donde se pueda ejecutar Red Hat OpenShift Container Platform, lo que permite evitar la dependencia de un solo proveedor.

RHOAR proporciona:

- Acceso a archivos binarios soportados y creados por Red Hat para marcos (frameworks) de desarrollo y tiempos de ejecución de microservicios seleccionados.

- Acceso a archivos binarios soportados y creados por Red Hat para módulos de integración que reemplazan o mejoran la implementación de un marco (framework) de un patrón de microservicios para usar funciones de OpenShift.
- Compatibilidad para que los desarrolladores puedan escribir aplicaciones con los marcos de desarrollo de microservicios, los tiempos de ejecución y los módulos de integración seleccionados, así como con las integraciones con los servicios externos seleccionados, como servidores de bases de datos.
- Compatibilidad de producción para implementar aplicaciones con los marcos (frameworks) de desarrollo de microservicios, los tiempos de ejecución y los módulos de integración seleccionados, así como con las integraciones en un clúster OpenShift soportado.

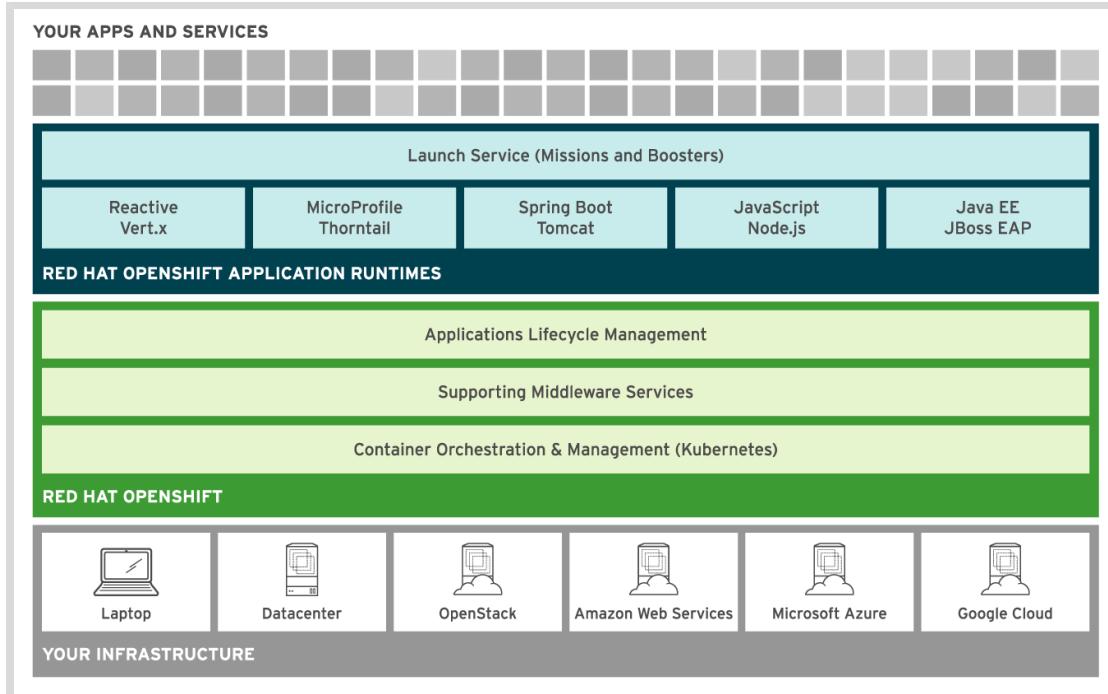


Figura 9.6: Arquitectura de Red Hat OpenShift Application Runtimes

Red Hat admite los siguientes marcos (frameworks) de desarrollo y tiempos de ejecución populares como parte de RHOAR:

Thorntail

Implementación del estándar MicroProfile que compila aplicaciones nativas de la nube usando API de Java EE existentes y nuevas.

Eclipse Vert.x

Un marco (framework) de desarrollo reactivo y de baja latencia basado en E/S asíncronas y secuencias de eventos.

Spring Boot

Un marco de desarrollo nativo de la nube basado en el popular Spring Framework y la configuración automática.

Node.js

Un tiempo de ejecución de JavaScript que proporciona un modelo de E/S basado en eventos y operaciones sin bloqueo, lo que le permite escribir aplicaciones de alto rendimiento que son ligeras y eficientes.

**nota**

Para obtener más detalles sobre RHOAR, consulte Descripción general de Red Hat OpenShift Application Runtimes [<https://developers.redhat.com/products/rhoar/overview/>]

Complemento (plug-in) Fabric8 Maven

El complemento (plug-in) Fabric8 Maven es uno de los componentes del proyecto Fabric8 de código abierto. Fabric8 es una plataforma de desarrollo que soporta el ciclo de vida de desarrollo de software completo de aplicaciones nativas en la nube.

Puede usar el complemento (plug-in) Fabric8 Maven para implementar aplicaciones de Java en un clúster OpenShift usando una fuente de entrada de compilación binaria. Esto significa que el complemento (plug-in) compila y empaqueta la aplicación, genera la imagen de contenedor con el artefacto empaquetado y crea recursos OpenShift para implementar la aplicación en OpenShift.

Configuración del complemento (plug-in)

Para usar el complemento (plug-in) Fabric8 Maven con un proyecto, actualice el archivo de configuración de Maven `pom.xml` del proyecto para habilitar y configurar el complemento (plug-in). Debe agregar una entrada `plugin` a la lista `plugins` en la sección `build` del archivo `pom.xml`. La configuración que sigue es una configuración mínima para habilitar el complemento (plug-in) Fabric8 Maven para una aplicación Java:

```
<build>
  <plugins>
    <plugin>
      <groupId>io.fabric8</groupId> ①
      <artifactId>fabric8-maven-plugin</artifactId> ②
      <version>{fabric8.version}<version> ③
      <executions>
        <execution>
          <goals> ④
            <goal>resource</goal>
            <goal>build</goal>
          </goals>
        </execution>
      </executions>
      <!-- additional configuration here --> ⑤
    </plugin>
  </plugins>
</build>
```

- ① ② Especifica el ID del artefacto y grupo de complemento (plug-in) Fabric8 Maven. Maven usa estos valores para localizar y descargar el complemento (plug-in).
- ③ Determina la versión del complemento (plug-in) que Maven descarga y usa. Puede usar una comunidad o una versión de Red Hat del complemento (plug-in).

En este ejemplo, el valor de la etiqueta `version` es `.fabric8.version`. Los caracteres `{}` indican que `fabric8.version` es propiedad de Maven. Puede encontrar el valor de la propiedad `fabric8.version` en la sección `properties` del archivo `pom.xml`.

- ④ Configura el complemento (plug-in) para ejecutar el recurso y las metas de compilación con el comando `mvn install`.
- ⑤ Existen otras opciones de configuración detallada para el complemento (plug-in) Fabric8 Maven. Puede usar las opciones para personalizar, entre otras cosas, la configuración de los recursos de OpenShift necesarios para la aplicación. Si no se proporcionan opciones adicionales, el complemento (plug-in) Fabric8 Maven crea un conjunto de recursos OpenShift simples para la aplicación.

Como alternativa, puede usar fragmentos YAML de Fabric8 para personalizar los recursos de OpenShift para la aplicación. En este curso, usará fragmentos YAML de Fabric8 para crear recursos OpenShift personalizados, que se describen más adelante.

Agregue este segmento XML al archivo `pom.xml` del proyecto para agregar el complemento (plug-in) de Fabric Maven. Consulte las referencias al final de esta presentación para obtener más detalles sobre la configuración del complemento (plug-in) Fabric8 Maven.

Metas de complementos (plug-ins)

Un objetivo del complemento (plug-in) de Maven representa una tarea bien definida en el proceso del ciclo de vida del desarrollo de software. Ejecute un objetivo del complemento (plug-in) de Maven con el comando `mvn`:

```
[user@host sample-app]$ mvn <plug-in goal name>
```

El complemento (plug-in) Fabric8 Maven proporciona un conjunto de objetivos para hacer frente al desarrollo de aplicaciones Java nativas de la nube:

`fabric8:resource`

Crea descriptores de recursos Kubernetes y OpenShift. El complemento (plug-in) almacena todos los descriptores generados en el subdirectorio `target/classes/META-INF/fabric8` del proyecto.

`fabric8:build`

Compila y empaqueta la aplicación Java para crear un artefacto binario y, a continuación, usa el artefacto para compilar la imagen de contenedor de aplicación asociada.

El complemento (plug-in) usa *generadores* para detectar automáticamente los parámetros de compilación necesarios para compilar y empaquetar la aplicación. De particular importancia, los generadores identifican una imagen de generador adecuada para usar para la aplicación. Para aplicaciones Java genéricas, la imagen de constructor predeterminada es `fabric8/s2i-java` (comunidad) o `jboss-fuse-6/fis-java Openshift` (Red Hat), dependiendo de la versión del complemento (plug-in) Fabric8 Maven.

Si el complemento (plug-in) detecta el acceso a un clúster OpenShift, inicia una compilación binaria de OpenShift. El complemento (plug-in) admite compilaciones binarias de origen a imagen y Docker. De forma predeterminada, el complemento (plug-in) ejecuta una estrategia de compilación de origen a imagen.

Para las compilaciones de OpenShift, el complemento (plug-in) compila una configuración de compilación de aplicaciones y un recurso de flujo de imagen en el clúster OpenShift. Para las estrategias de compilación de Source y Docker, el complemento (plug-in) actualiza la secuencia de imágenes con la nueva imagen de contenedor.

`fabric8:apply`

Aplica los recursos generados al clúster Kubernetes u OpenShift conectado.

fabric8:resource-apply

Ejecute las metas `fabric8:resource` y `fabric8:apply`.

fabric8:deploy

Ejecuta las metas `fabric8:resource`, `fabric8:build` y `fabric8:apply`.

fabric8:undeploy

Quita los recursos del clúster OpenShift.

El complemento (plug-in) Fabric8 Maven admite otros objetivos que están fuera del alcance de este curso. Consulte las referencias al final de esta presentación para obtener más detalles sobre los objetivos.

Personalización de recursos OpenShift

Con una configuración de proyecto mínima, el complemento (plug-in) Fabric8 Maven genera un conjunto de recursos OpenShift para admitir la implementación de la aplicación en OpenShift. En algunos escenarios, los recursos de OpenShift generados pueden no ser suficientes para la implementación de OpenShift. Personalice los recursos generados de una de estas dos maneras: agregue fragmentos YAML de recursos OpenShift al subdirectorio `src/main/fabric8` del proyecto o agregue configuración adicional al archivo `pom.xml` de proyecto. En este curso, usará fragmentos de recursos YAML para personalizar la configuración de OpenShift para un proyecto.

Si agrega solo la configuración mínima de Fabric8 al proyecto, el complemento (plug-in) crea un recurso de configuración de implementación y servicio para la aplicación. Si la imagen de contenedor asociada expone un puerto, el complemento (plug-in) también crea un recurso de ruta para exponer el servicio. El complemento (plug-in) escribe cada definición de recurso en un archivo del directorio `target/classes/META-INF/fabric8/openshift`. Todas estas definiciones de recursos se combinan en un archivo `openshift.yml`, en el subdirectorio `target/classes/META-INF/fabric8` del proyecto.

El complemento (plug-in) también procesa archivos de fragmentos YAML en el directorio `src/main/fabric8`. El complemento (plug-in) usa el contenido de cada fragmento para invalidar la definición de recurso predeterminada correspondiente. El contenido de cada archivo imita la estructura de un recurso OpenShift, pero omite cualquier información que no haya cambiado. Estos archivos están diseñados para ser pequeños y compactos, y representan solo la configuración que debe cambiar de la configuración de recursos predeterminada.

Cada archivo de fragmento sigue una convención de nomenclatura de archivos. El complemento (plug-in) usa el nombre de archivo de fragmento para identificar el recurso OpenShift que se debe invalidar. Los nombres de archivo de fragmento siguen el patrón `[name]-type.yml`.

El valor `name` es opcional y representa el nombre del recurso. Si no se proporciona un nombre, el complemento (plug-in) usa el nombre de la aplicación como nombre del recurso.

El valor `type` corresponde al tipo de recurso OpenShift que modifica este fragmento. Debe ser uno de los siguientes:

| Tipo | Nombre de archivo |
|--------------------------------|---------------------------|
| Servicio | <code>svc, service</code> |
| Ruta | <code>route</code> |
| Deployment (Implementación) | <code>deployment</code> |

| | |
|------------------|----------------------|
| DeploymentConfig | dc, deploymentconfig |
| ConfigMap | cm, configmap |
| Secreto | secret |

Por ejemplo, considere el contenido de `src/main/fabric8/route.yml` a continuación:

```
spec:
  host: app.alternate.com
```

Este fragmento cambia el nombre de host predeterminado de la ruta de la aplicación a `app.alternate.com`.

Red Hat Middleware para OpenShift

Red Hat ofrece un conjunto de imágenes de contenedor de middleware que le permite implementar las aplicaciones de middleware Red Hat en OpenShift.

Por ejemplo, use el Red Hat Java S2I para la imagen de contenedor de OpenShift para implementar una aplicación empaquetada como *archivo fat JAR*. Una colección de archivos fat jar es una aplicación ejecutable que contiene todos los recursos y las clases en un paquete, incluidas las dependencias de tiempo de ejecución, en el mismo archivo jar.

El Catálogo del contenedor de Red Hat incluye siguientes imágenes de los siguientes productos:

- Red Hat JBoss EAP para OpenShift: le permite compilar e implementar una aplicación usando Red Hat JBoss Enterprise Application Platform.
- Red Hat JBoss AMQ para OpenShift: le permite implementar un agente de mensajería MQ basado en Apache ActiveMQ.
- Red Hat JBoss Fuse Integration Services para OpenShift: ofrece un conjunto de herramientas e imágenes en contenedores que permiten el desarrollo, la implementación y la administración de microservicios de integración dentro de OpenShift.
- Red Hat OpenShift Service Mesh: permite a los equipos de operaciones y a los desarrolladores proporcionar monitoreo del tráfico, control de acceso, descubrimiento, seguridad y resistencia a un grupo de servicios.
- Red Hat 3scale API Management: comparta, proteja, distribuya, controle y monetize API.

Las imágenes del compilador para otros productos de middleware de Red Hat también están disponibles. Consulte las referencias al final de esta presentación para obtener más información acerca de estas y otras imágenes de contenedor.



Referencias

Para obtener más información sobre Cloud Native Computing Foundation, consulte <https://www.cncf.io/>

Para obtener más información acerca de RHOAR, consulte <https://developers.redhat.com/products/rhoar/overview/>

La documentación de código abierto para el complemento (plug-in) Fabric8 Maven está disponible en <https://maven.fabric8.io/>

Para obtener más información acerca del complemento (plug-in) Fabric8 Maven, consulte el apéndice *Complemento (plug-in) Fabric8 Maven* de la guía *Fuse on OpenShift* en la documentación del producto Red Hat JBoss Fuse 7.3 en https://access.redhat.com/documentation/en-us/red_hat_fuse/7.3/html-single/fuse_on_openshift_guide/index#fabric8-maven-plugin

Para obtener más información acerca de la configuración de Maven para usar dependencias soportadas, consulte la sección *Prepare su entorno de desarrollo* del capítulo *Introducción para desarrolladores* de la guía *Fuse en OpenShift* de la documentación del producto Red Hat Fuse 7.3 en https://access.redhat.com/documentation/en-us/red_hat_fuse/7.3/html-single/fuse_on_openshift_guide/index#get-started-prepare-dev-env

► Ejercicio Guiado

Implementación de una aplicación con Red Hat OpenShift Application Runtimes

En este ejercicio, usará el complemento (plug-in) Fabric8 Maven para implementar un microservicio en el clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Configure un recurso de configuración de implementación OpenShift personalizado mediante el complemento (plug-in) Fabric8 Maven.
- Configure un recurso de mapa de configuración de OpenShift personalizado mediante el complemento (plug-in) Fabric8 Maven.
- Implemente un microservicio mediante el complemento (plug-in) Fabric8 Maven.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación de microservicio en el repositorio Git (`micro-java`).

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab micro-java start
```

► 1. Inspeccione el código fuente de Java de la aplicación de muestra `micro-java`.

- Escriba el subdirectorio `micro-java` de su clon local del repositorio Git D0288-apps. Compruebe la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd ~/D0288-apps/micro-java
[student@workstation micro-java]$ git checkout master
...output omitted...
```

- Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation micro-java]$ git checkout -b micro-config
Switched to a new branch 'micro-config'
[student@workstation micro-java]$ git push -u origin micro-config
...output omitted...
* [new branch]      micro-config -> micro-config
Branch micro-config set up to track remote branch micro-config from origin.
```

- 1.3. Revise el archivo de código fuente `JaxRsActivator.java` en el subdirectorio `src/main/java/com/redhat/training/openshift/hello` del código fuente `micro-java`:

```
...output omitted...
@ApplicationPath("/api")
public class JaxRsActivator extends Application {
}
...output omitted...
```

El código fuente especifica que se tiene acceso a la aplicación en una ruta de acceso de `/api`.

- 1.4. Revise el archivo de código fuente `HelloResource.java` en el subdirectorio `src/main/java/com/redhat/training/openshift/hello` del código fuente `micro-java`:

```
package com.redhat.training.openshift.hello;

import javax.inject.Inject;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

@Path("/")
public class HelloResource❶ {

    @GET
    @Path("/hello")❷
    @Produces("text/plain")
    public String hello() {
        String hostname = System.getenv().getOrDefault("HOSTNAME", "unknown");❸
        String message = System.getenv().getOrDefault("APP_MSG", null);❹
        String response = "";

        if (message == null) {
            response = "Hello world from host "+hostname+"\n";❺
        } else {
            response = "Hello world from host ["+hostname+"].\n";
            response += "Message received = "+message+"\n";❻
        }
        return response;
    }
}
```

❶ Esta clase define un recurso REST para la aplicación.

- ❷ Dado que tiene acceso a la aplicación en /api, tiene acceso a este recurso de aplicación en /api/hello.
- ❸❹ La aplicación usa las variables de entorno HOSTNAME y APP_MSG en la aplicación.
- ❺ Si la variable de entorno APP_MSG no está definida, el recurso de aplicación responde con un mensaje que contiene el nombre de host del servidor en el que se ejecuta la aplicación.
- ❻ Si se define la variable de entorno APP_MSG, el mensaje de respuesta contiene el valor de las variables de entorno HOSTNAME y APP_MSG.

▶ 2. Revise la configuración del complemento (plug-in) Fabric8 Maven en el archivo de proyecto pom.xml:

- 2.1. Revise los atributos de nivel de proyecto del pom.xml, que se encuentra cerca de la parte superior del archivo:

```
<?xml version="1.0" encoding="UTF-8"?>
...output omitted...
<groupId>com.redhat.training.openshift</groupId>
<artifactId>hello</artifactId>❶
<version>1.0</version>❷
...output omitted...
<properties>
...output omitted...
    <!-- Thorntail dependency versions -->
    <version.thorntail>2.4.0.Final</version.thorntail>❸

    <!-- other plugin versions -->
...output omitted...
    <version.fabric8.plugin>4.1.0</version.fabric8.plugin>❹
...output omitted...
</properties>
...output omitted...
```

- ❶❷ El nombre y la versión de la aplicación. El complemento (plug-in) Fabric8 Maven crea un recurso de etiqueta de secuencia de imagen a partir de estos valores, hello:1.0.
- ❸❹ Estas propiedades de versión establecen la versión del tiempo de ejecución de la aplicación Thorntail y el complemento (plug-in) Fabric8 Maven, respectivamente. Se hace referencia a estas propiedades más abajo en el archivo pom.xml donde se configuran los complementos (plug-ins) Thorntail y Fabric Maven.

- 2.2. Revise la lista de complementos (plug-ins) en la sección de build de pom.xml, cerca del final del archivo:

```
...output omitted...
<build>
...output omitted...
    <plugins>
...output omitted...
        <plugin>❶
            <groupId>io.thorntail</groupId>
```

```

<artifactId>thorntail-maven-plugin</artifactId>
<version>${version.thorntail}</version>
...output omitted...
</plugin>
<plugin>❷
  <groupId>io.fabric8</groupId>
  <artifactId>fabric8-maven-plugin</artifactId>
  <version>${version.fabric8.plugin}</version>
  <executions>
    <execution>
      <id>fmp</id>
      <goals>
        <goal>resource</goal>
        <goal>build</goal>
      </goals>
    </execution>
  </executions>
</plugin>
</plugins>
</build>
...output omitted...

```

❶❷ Se configuran dos complementos (plug-ins) para el proyecto.

- ❶ El complemento (plug-in) Thorntail Maven empaqueta el código fuente como una aplicación Thorntail. La versión del complemento (plug-in) viene determinada por el valor de la propiedad `version.thorntail`, que se establece en la sección `properties` del archivo `pom.xml`.
- ❷ El complemento (plug-in) Fabric8 Maven crea recursos OpenShift para la aplicación y compila una imagen de contenedor a partir del código empaquetado por Thorntail. La versión del complemento (plug-in) viene determinada por el valor de la propiedad `version.fabric8.plugin`, que además se establece en la sección de las propiedades del archivo `pom.xml`.

► 3. Genere recursos de OpenShift para la aplicación.

- 3.1. Desde el directorio del proyecto, ejecute el comando `mvn clean` para quitar los artefactos de compilación de compilaciones anteriores. A continuación, genere los recursos de OpenShift con el objetivo `fabric8:resource` de Maven:

```

[student@workstation micro-java]$ mvn clean
...output omitted...
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
...output omitted...
[student@workstation micro-java]$ mvn fabric8:resource
[INFO] Scanning for projects...
...output omitted...
[INFO] --- fabric8-maven-plugin:4.1.0:resource (default-cli) @ hello ---
...output omitted...
[INFO] F8: fmp-controller: Adding a default DeploymentConfig
[INFO] F8: fmp-service: Adding a default service 'hello' with ports [8080]
[INFO] F8: fmp-revision-history: Adding revision history limit to 2
[INFO] F8: validating .../fabric8/openshift/hello-service.yml ...❶

```

```
[INFO] F8: validating .../fabric8/openshift/hello-deploymentconfig.yml ...
[INFO] F8: validating .../fabric8/openshift/hello-route.yml resource
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
...output omitted...
```

- ① El objetivo `fabric8:resource` Maven crea tres archivos YAML. En conjunto, estos archivos definen un servicio, una configuración de implementación y un recurso de ruta para la aplicación de muestra. Estos archivos se encuentran en el subdirectorio `target/classes/META-INF/fabric8/openshift` del proyecto.
- 3.2. Revise el archivo `hello-deploymentconfig.yml` generado en el directorio `target/classes/META-INF/fabric8/openshift`.

```
---
apiVersion: apps.openshift.io/v1
kind: DeploymentConfig
metadata:
...output omitted...
  name: hello
spec:
  replicas: 1
...output omitted...
  template:
    spec:
      containers:
        - env:
            - name: KUBERNETES_NAMESPACE
              valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
        image: hello:1.0
...output omitted...
```

El archivo define una configuración de implementación denominada `hello` que implementa un único contenedor mediante una imagen de la etiqueta de secuencia de imágenes `hello:1.0`.

- 3.3. Revise el archivo `hello-service.yml` generado en el directorio `target/classes/META-INF/fabric8/openshift`.

```
---
apiVersion: v1
kind: Service
metadata:
...output omitted...
  name: hello
spec:
  ports:
    - name: http
      port: 8080
      protocol: TCP
```

```
targetPort: 8080
selector:
  app: hello
  provider: fabric8
  group: com.redhat.training.openshift
```

Este archivo define un servicio denominado **hello** para los pods de aplicación definidos en la configuración de implementación **hello**.

- 3.4. Revise el archivo **hello-route.yml** generado en el directorio **target/classes/META-INF/fabric8/openshift**.

```
...
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  ...output omitted...
  name: hello
spec:
  port:
    targetPort: 8080
  to:
    kind: Service
    name: hello
```

Este archivo define un recurso de ruta denominado **hello** que expone el servicio **hello**.

- 3.5. La meta **fabric8:resource** Maven también genera una lista combinada de todos los recursos. Revise el archivo **openshift.yml** generado en el subdirectorio **target/classes/META-INF/fabric8** del proyecto:

```
...output omitted...
kind: "List"
...output omitted...
  kind: "Service"
...output omitted...
  kind: "DeploymentConfig"
...output omitted...
  kind: "Route"
...output omitted...
```

▶ 4. Compile e implemente la aplicación.

- 4.1. 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 micro-java]$ source /usr/local/etc/ocp4.config
```

- 4.2. Inicie sesión en OpenShift con su nombre de usuario de desarrollador:

```
[student@workstation micro-java]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

- 4.3. Cree un nuevo proyecto para la aplicación. Coloque como prefijo del nombre del proyecto el nombre de usuario de desarrollador.

```
[student@workstation micro-java]$ oc new-project ${RHT_OCP4_DEV_USER}-micro-java
Now using project "youruser-micro-java" on server ...
...output omitted...
```

- 4.4. Compile e implemente la aplicación con el complemento (plug-in) Fabric8 Maven: monitoree la salida para comprobar que la compilación se ejecuta en modo OpenShift y que se crearon recursos de OpenShift.

```
[student@workstation micro-java]$ mvn fabric8:deploy
[INFO] Scanning for projects...
...output omitted...
[INFO] --- fabric8-maven-plugin:4.1.0:resource (fmp) @ hello ---
...output omitted...
[INFO] --- fabric8-maven-plugin:4.1.0:build (fmp) @ hello ---❶
[INFO] F8: Running in OpenShift mode
[INFO] F8: Using OpenShift build with strategy S2I
[INFO] F8: Running generator thorntail-v2
...output omitted...
[INFO] --- fabric8-maven-plugin:4.1.0:deploy (default-cli) @ hello ---❷
[INFO] F8: Using OpenShift ... with manifest .../openshift.yml
[INFO] OpenShift platform detected
[INFO] F8: Using project: youruser-micro-java❸
[INFO] F8: Creating a Service from openshift.yml❹ ... name hello
...output omitted...
[INFO] F8: Creating a DeploymentConfig from openshift.yml name hello
...output omitted...
[INFO] F8: Creating Route youruser-micro-java:hello host: null
[INFO] F8: HINT: Use the command `oc get pods -w` to watch your pods start up
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
...output omitted...
```

- ❶ Comienza la fase de compilación. En esta fase, el complemento (plug-in) usa una compilación binaria de origen a imagen para crear una imagen de contenedor para la aplicación.
- ❷ Comienza la fase de implementación.
- ❸ El complemento (plug-in) detecta el proyecto OpenShift activo. Todos los recursos se crean en este proyecto.
- ❹ El complemento (plug-in) usa `openshift.yml` para crear un servicio OpenShift, la configuración de implementación y el recurso de ruta.

- 5. Revise los recursos creados por el complemento (plug-in) Fabric8 Maven.

```
[student@workstation micro-java]$ oc status
In project youruser-micro-java on server ...

http://hello-youruser-micro-java... to pod port 8080 (svc/hello)
dc/hello deploys istag/hello:1.0 <- bc/hello-s2i source builds ...
  deployment #1 deployed 2 minutes ago - 1 pod
...output omitted...
```

Existe un recurso de configuración de implementación, ruta y servicio en el proyecto, cada uno con un nombre de hello. También existe una configuración de compilación y un recurso de etiqueta de secuencia de imágenes en el proyecto.

► 6. Pruebe la aplicación.

- 6.1. Espere a que la aplicación esté lista y en ejecución:

```
[student@workstation micro-java]$ oc get pods
NAME        READY   STATUS    RESTARTS   AGE
hello-1-5pw6q   1/1     Running   1          14m
hello-1-deploy   0/1     Completed  0          14m
hello-s2i-1-build 0/1     Completed  0          16m
```

- 6.2. Pruebe el acceso externo a la aplicación. Recuerde agregar /api/hello al final de la dirección URL de ruta para tener acceso al recurso REST HelloResource para la aplicación.

```
[student@workstation D0288-apps]$ ROUTE_URL=$(oc get route \
> hello --template='{{.spec.host}}')
[student@workstation D0288-apps]$ curl ${ROUTE_URL}/api/hello
Hello world from host hello-1-5pw6q
```

Dado que la configuración de implementación de la aplicación no define un valor para la variable de entorno ADD_MSG, la salida solo contiene el nombre de host del contenedor.

► 7. Actualice el proyecto para implementar pods de aplicación con un valor para la variable de entorno APP_MSG.

- 7.1. Revise el archivo de fragmento cm.yaml generado en el directorio ~/D0288/labs/micro-java:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: env-config
data:
  APP_MSG: sample external configuration
```

El fragmento YAML anterior define un recurso de mapa de configuración con un nombre de env-config. En este mapa de configuración, se define una variable APP_MSG con un valor de sample external configuration.

- 7.2. Revise el archivo de fragmento deployment.yaml generado en el directorio ~/D0288/labs/micro-java:

```
spec:
  template:
    spec:
      containers:
        - envFrom:
          - configMapRef:
            name: env-config
```

El complemento (plug-in) Fabric8 Maven usa este fragmento para actualizar la configuración de implementación predeterminada que genera. En particular, cada contenedor implementado se inyecta con variables de entorno de la asignación de configuración `env-config`.

- 7.3. Copie los archivos de fragmento en el subdirectorio `src/main/fabric8` de la aplicación.

```
[student@workstation micro-java]$ cp -v ~/DO288/labs/micro-java/*.yml \
> ./src/main/fabric8/
'...labs/micro-java/cm.yml' -> './src/main/fabric8/cm.yml'
'...labs/micro-java/deployment.yml' -> './src/main/fabric8/deployment.yml'
```

- 7.4. Confirme los fragmentos YAML en la bifurcación `micro-config`:

```
[student@workstation micro-java]$ git add src/main/fabric8/*.yml
[student@workstation micro-java]$ git commit -am "Add YAML fragments."
...output omitted...
```

- 8. Vuelva a implementar la aplicación. Compruebe que un complemento (plug-in) Fabric8 Maven crea un recurso de mapa de configuración. Compruebe que la aplicación responde con el valor de la variable `APP_MSG` del mapa de configuración.

- 8.1. Vuelva a implementar la aplicación con el complemento (plug-in) Fabric8 Maven:

```
[student@workstation micro-java]$ mvn fabric8:deploy
...output omitted...
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
...output omitted...
```

- 8.2. Verifique la presencia del mapa de configuración `env-config`:

```
[student@workstation micro-java]$ oc get cm/env-config
NAME          DATA   AGE
env-config     1      84m
```

- 8.3. Espere a que se implementen los nuevos pods de la aplicación. Cuando la salida de `oc get pods` indica que una nueva implementación está lista y en ejecución, continúe con el paso siguiente:

```
[student@workstation micro-java]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
hello-1-deploy 0/1     Completed  0          12m
hello-3-deploy 0/1     Completed  0          117s
hello-3-n2rn2   1/1     Running   0          108s
hello-s2i-1-build 0/1     Completed  0          15m
hello-s2i-2-build 0/1     Completed  0          4m42s
```

8.4. Verifique que la respuesta de la aplicación incluya el valor de la variable APP_MSG:

```
[student@workstation micro-java]$ curl ${ROUTE_URL}/api/hello
Hello world from host [hello-3-m9k4j].
Message received = sample external configuration
```



nota

Si el comando curl anterior devuelve una página HTML de error que indica que la aplicación no está disponible, probablemente esto significa que el contenedor de la aplicación aún no está listo para responder a las solicitudes. Espere unos segundos y vuelva a probar el mismo comando.

Este tipo de problema se puede evitar si agrega un sondeo de disponibilidad a la configuración de implementación.

- ▶ 9. Limpieza: Elimine los recursos creados durante este ejercicio.

```
[student@workstation micro-java]$ oc delete project \
> ${RHT_OCP4_DEV_USER}-micro-java
```

Finalizar

En workstation, ejecute el script lab micro-java finish para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes. La acción de finalizar lanza este proyecto y sus recursos.

```
[student@workstation ~]$ lab micro-java finish
```

Esto concluye el ejercicio guiado.

► Trabajo de laboratorio

Compilación de aplicaciones nativas de la nube para OpenShift

Lista de verificación de rendimiento

En este trabajo de laboratorio, usará el complemento (plug-in) Fabric8 Maven para implementar una aplicación en OpenShift que se comunica con una base de datos que se ejecuta fuera del clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Cree un servicio de base de datos para la aplicación que apunta al servidor de la base de datos MariaDB fuera del clúster OpenShift.
- Configure los recursos OpenShift personalizados mediante fragmentos YAML de Fabric8.
- Implemente la aplicación de backend To Do List usando el complemento (plug-in) Fabric8 Maven.

Andes De Comenzar

Para realizar este ejercicio, necesita acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación de muestra todo-api-micro en el repositorio Git.
- El servidor de base de datos MariaDB externo.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos, completar la base de datos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab todo-migrate start
```

Requisitos

El equipo de desarrollo de back-end To Do List está migrando la aplicación Thorntail a OpenShift. El código fuente de la aplicación se encuentra en el subdirectorio `todo-api-micro` del repositorio Git clonado.

Debe configurar la aplicación e implementarla en un clúster OpenShift, según los siguientes requisitos:

- El nombre del proyecto es `youruser-todo-migrate`. El usuario developer debe ser el propietario del proyecto.
- Existe un servicio `tododb` en el proyecto OpenShift que se conecta a una base de datos externa. Para obtener el FQDN de la base de datos externa, reemplace las aplicaciones del dominio comodín del clúster OpenShift por `mysql`. Por ejemplo, si el dominio comodín

del clúster es `apps.cluster.domain.example.com`, el dominio de base de datos es `mysql.cluster.domain.example.com`.

- Se crea un recurso de mapa de configuración de OpenShift como parte de la implementación.

El mapa de configuración define las variables necesarias para acceder a la base de datos externa:

- `DATABASE_USER`: `todoapp`
- `DATABASE_PASSWORD`: `redhat123`
- `DATABASE_SVC_HOSTNAME`: `tododb`
- `DATABASE_NAME`: `todo`

Use un fragmento de YAML de Fabric8 para crear el recurso de mapa de configuración.



Importante

Para simplificar el trabajo de laboratorio, defina la contraseña de la base de datos en el recurso de mapa de configuración. Una mejor práctica es definir la contraseña de la base de datos en un recurso secreto.

- Se crea un recurso de configuración de OpenShift personalizado como parte de la implementación.

Use un fragmento de YAML de Fabric8 para agregar todas las variables del recurso de mapa de configuración al contenedor de aplicaciones como variables de entorno.

- Confirme todos los cambios al repositorio Git remoto.

Para probar una implementación correcta, el extremo (endpoint) `todo/api/items/6` de la aplicación debe devolver datos JSON.

Pasos

1. Verifique la conectividad al servidor de base de datos externa.
2. Cree un servicio OpenShift denominado `tododb` que se conecte a la instancia de base de datos externa. El servicio debe crearse en el proyecto `youruser-todo-migrate`.
3. Cree una bifurcación `todo-migrate` de la bifurcación `master` en el clon local del repositorio `D0288-apps`. Pase al subdirectorio `todo-api-micro` del proyecto.
4. Compile e implemente la aplicación. Compruebe que el pod de la aplicación no se puede implementar porque las variables de entorno necesarias no están definidas.
5. Cree el fragmento YAML que Fabric8 usa para generar el recurso de mapa de configuración personalizado que define las variables de entorno de base de datos.
Puede elegir definir las variables de entorno necesarias en un recurso de mapa de configuración personalizado o directamente en la configuración de implementación.
6. Cree el fragmento YAML que Fabric8 usa para generar el recurso de configuración de implementación personalizado.
7. Aplique el mapa de configuración personalizado y el recurso de configuración de implementación al proyecto. Compruebe que la aplicación se implemente sin errores.

Use la ruta externa para probar el acceso al recurso de aplicación `todo/api/items/6`. Una respuesta correcta devuelve datos JSON.

8. Una vez que las pruebas de la aplicación se realicen correctamente, confirme e inserte los cambios de código en el repositorio remoto.

Evaluación

Inicie sesión con el usuario `student` en la máquina `workstation` y use el comando `lab` para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

```
[student@workstation ~]$ lab todo-migrate grade
```

Finalizar

En `workstation`, ejecute el comando `lab todo-migrate finish` para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab todo-migrate finish
```

Esto concluye el trabajo de laboratorio.

► Solución

Compilación de aplicaciones nativas de la nube para OpenShift

Lista de verificación de rendimiento

En este trabajo de laboratorio, usará el complemento (plug-in) Fabric8 Maven para implementar una aplicación en OpenShift que se comunica con una base de datos que se ejecuta fuera del clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Cree un servicio de base de datos para la aplicación que apunta al servidor de la base de datos MariaDB fuera del clúster OpenShift.
- Configure los recursos OpenShift personalizados mediante fragmentos YAML de Fabric8.
- Implemente la aplicación de backend To Do List usando el complemento (plug-in) Fabric8 Maven.

Andes De Comenzar

Para realizar este ejercicio, necesita acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La aplicación de muestra todo-api-micro en el repositorio Git.
- El servidor de base de datos MariaDB externo.

Ejecute el siguiente comando en **workstation** para validar los requisitos previos, completar la base de datos y descargar los archivos requeridos para completar este ejercicio:

```
[student@workstation ~]$ lab todo-migrate start
```

Requisitos

El equipo de desarrollo de back-end To Do List está migrando la aplicación Thorntail a OpenShift. El código fuente de la aplicación se encuentra en el subdirectorio **todo-api-micro** del repositorio Git clonado.

Debe configurar la aplicación e implementarla en un clúster OpenShift, según los siguientes requisitos:

- El nombre del proyecto es **youruser-todo-migrate**. El usuario developer debe ser el propietario del proyecto.
- Existe un servicio **tododb** en el proyecto OpenShift que se conecta a una base de datos externa. Para obtener el FQDN de la base de datos externa, reemplace las aplicaciones del dominio comodín del clúster OpenShift por **mysql**. Por ejemplo, si el dominio comodín

del clúster es `apps.cluster.domain.example.com`, el dominio de base de datos es `mysql.cluster.domain.example.com`.

- Se crea un recurso de mapa de configuración de OpenShift como parte de la implementación.

El mapa de configuración define las variables necesarias para acceder a la base de datos externa:

- `DATABASE_USER`: `todoapp`
- `DATABASE_PASSWORD`: `redhat123`
- `DATABASE_SVC_HOSTNAME`: `tododb`
- `DATABASE_NAME`: `todo`

Use un fragmento de YAML de Fabric8 para crear el recurso de mapa de configuración.



Importante

Para simplificar el trabajo de laboratorio, defina la contraseña de la base de datos en el recurso de mapa de configuración. Una mejor práctica es definir la contraseña de la base de datos en un recurso secreto.

- Se crea un recurso de configuración de OpenShift personalizado como parte de la implementación.

Use un fragmento de YAML de Fabric8 para agregar todas las variables del recurso de mapa de configuración al contenedor de aplicaciones como variables de entorno.

- Confirme todos los cambios al repositorio Git remoto.

Para probar una implementación correcta, el extremo (endpoint) `todo/api/items/6` de la aplicación debe devolver datos JSON.

Pasos

1. Verifique la conectividad al servidor de base de datos externa.

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

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

- 1.2. Determine el nombre de host del servidor de base de datos externa.

```
[student@workstation ~]$ MYSQL_DB=$(echo \
> mysql.ocp-$${RHT_OCP4_WILDCARD_DOMAIN#"apps."})
```

- 1.3. Conéctese con la base de datos MySQL externa.

```
[student@workstation ~]$ mysql -h${MYSQL_DB} -utodoapp -predhat123 todo
Reading table information for completion of table and column names
...output omitted...
MariaDB [todo]>
```

- 1.4. Salga del cliente MySQL para volver al prompt de shell.

```
MariaDB [todo]> exit
Bye
[student@workstation ~]$
```

2. Cree un servicio OpenShift denominado tododb que se conecte a la instancia de base de datos externa. El servicio debe crearse en el proyecto *youruser-todo-migrate*.

- 2.1. Inicie sesión en OpenShift con su cuenta de usuario de desarrollador:

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```

- 2.2. Cree un nuevo proyecto para alojar la aplicación:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-todo-migrate
```

- 2.3. Cree un servicio basado en un nombre externo:

```
[student@workstation ~]$ oc create service externalname tododb \
> --external-name ${MYSQL_DB}
service "tododb" created
```

- 2.4. Verifique que el servicio tododb exista y muestre una IP externa, pero ninguna IP de clúster:

```
[student@workstation ~]$ oc get svc
NAME      TYPE           ...   EXTERNAL-IP          PORT(S)      AGE
tododb    ExternalName   ...   mysql.cluster.domain.example.com <none>     6s
```

3. Cree una bifurcación todo-migrate de la bifurcación master en el clon local del repositorio D0288-apps. Pase al subdirectorio todo-api-micro del proyecto.

```
[student@workstation ~]$ cd ~/D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
[student@workstation D0288-apps]$ git checkout -b todo-migrate
Switched to a new branch 'todo-migrate'
[student@workstation D0288-apps]$ git push -u origin todo-migrate
...output omitted...
[student@workstation D0288-apps]$ cd todo-api-micro
[student@workstation todo-api-micro]$
```

4. Compile e implemente la aplicación. Compruebe que el pod de la aplicación no se puede implementar porque las variables de entorno necesarias no están definidas.

- 4.1. Despliegue la aplicación.

```
[student@workstation todo-api-micro]$ mvn fabric8:deploy
```

- 4.2. Despu s de implementar la aplicaci n, monitoree los registros del pod de la aplicaci n.

```
[student@workstation todo-api-micro]$ oc get pods
NAME           READY   STATUS      RESTARTS   AGE
todo-api-1-deploy  0/1     Completed   0          5m43s
todo-api-1-hj5hn  0/1     CrashLoopBackOff  2          90s
todo-api-s2i-1-build 0/1     Completed   0          7m12s
```

El valor exacto de STATUS (Estado) del pod puede variar, pero aumenta el n mero de RESTARTS (Reinicios).

```
[student@workstation todo-api-micro]$ oc logs -f todo-api-1-hj5hn
...output omitted...
... ERROR [...] ... ((("system-property" => "thorntail.datasources.datasources.MySQLDS.connection-url")) - failure
description: "WFLYCTL0211: Cannot resolve expression
'jdbc:mysql://${env.DATABASE_SVC_HOSTNAME}:3306/${env.DATABASE_NAME}'"
```

Se produce un error porque DATABASE_SVC_HOSTNAME y DATABASE_NAME no est n definidas.

5. Cree el fragmento YAML que Fabric8 usa para generar el recurso de mapa de configuraci n personalizado que define las variables de entorno de base de datos.

Puede elegir definir las variables de entorno necesarias en un recurso de mapa de configuraci n personalizado o directamente en la configuraci n de implementaci n.

- 5.1. Cree el archivo `src/main/fabric8/cm.yml` con el siguiente contenido:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: db-config
data:
  DATABASE_USER: todoapp
  DATABASE_PASSWORD: redhat123
  DATABASE_SVC_HOSTNAME: tododb
  DATABASE_NAME: todo
```

Como alternativa, est  disponible un archivo de fragmento YAML de soluci n en el directorio `/home/student/D0288/solutions/todo-migrate`. Puede copiarlo en el subdirectorio `src/main/fabric8`:

```
[student@workstation todo-api-micro]$ cp \
> ~/D0288/solutions/todo-migrate/cm.yml src/main/fabric8
```

6. Cree el fragmento YAML que Fabric8 usa para generar el recurso de configuraci n de implementaci n personalizado.

- 6.1. Cree el archivo `src/main/fabric8/deployment.yml` con el siguiente contenido:

```
spec:
  template:
    spec:
      containers:
        - envFrom:
          - configMapRef:
            name: db-config
```

El nombre de referencia del mapa de configuración debe coincidir con el nombre del recurso de mapa de configuración.

Como alternativa, está disponible un archivo de fragmento YAML de solución en el directorio /home/student/D0288/solutions/todo-migrate. Puede copiarlo en el subdirectorio src/main/fabric8:

```
[student@workstation todo-api-micro]$ cp \
> ~/D0288/solutions/todo-migrate/deployment.yml src/main/fabric8
```

7. Aplique el mapa de configuración personalizado y el recurso de configuración de implementación al proyecto. Compruebe que la aplicación se implemente sin errores. Use la ruta externa para probar el acceso al recurso de aplicación todo/api/items/6. Una respuesta correcta devuelve datos JSON.
- 7.1. Aplique los nuevos recursos de OpenShift al proyecto.

```
[student@workstation todo-api-micro]$ mvn fabric8:resource-apply
```

7.2. Revise los recursos creados por el complemento (plug-in) Fabric8 Maven.

```
[student@workstation todo-api-micro]$ oc describe dc/todo-api \
> | grep -A1 "Environment Variables"
  Environment Variables from:
    db-config ConfigMap Optional: false
```

El nombre del recurso del mapa de configuración debe coincidir con el nombre del recurso de mapa de configuración que contiene las variables de base de datos.

```
[student@workstation todo-api-micro]$ oc get configmap
NAME           DATA   AGE
db-config       4      5m16s
...output omitted...
[student@workstation todo-api-micro]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
...output omitted...
pod/todo-api-3-frhtt   1/1     Running   0          35s
...output omitted...
```

7.3. Espere a que la aplicación esté lista y en ejecución:

```
[student@workstation todo-api-micro]$ oc logs -f todo-api-3-frhtt
...output omitted...
... INFO [org.wildfly.swarm] (main) THORN99999: Thorntail is Ready
```

**nota**

Encontrará un error en los registros de la aplicación relacionado con SSL:

```
... ERROR ... Establishing SSL connection without server's identity
verification is not recommended. ...
```

Este error no causa errores funcionales en la aplicación. El error se produce porque la base de datos remota usa un certificado autofirmado para establecer una conexión SSL con la aplicación.

En este trabajo de laboratorio, puede ignorar este error.

- 7.4. Verifique que la aplicación obtenga datos de la base de datos externa.

Use el comando `curl` para probar que el recurso `todo/api/items/6` de aplicación devuelve datos JSON.

```
[student@workstation todo-api-micro]$ ROUTE_URL=$(oc get route todo-api \
> --template={{.spec.host}})
[student@workstation todo-api-micro]$ curl -s ${ROUTE_URL}/todo/api/items/6 \
> | python -m json.tool
{
    "description": "Verify that the To Do List application works",
...output omitted...
}
```

8. Una vez que las pruebas de la aplicación se realicen correctamente, confirme e inserte los cambios de código en el repositorio remoto.

```
[student@workstation todo-api-micro]$ git add src/main/fabric8/*
[student@workstation todo-api-micro]$ git commit -m "add YAML fragments"
...output omitted...
[student@workstation todo-api-micro]$ git push origin todo-migrate
...output omitted...
```

Evaluación

Inicie sesión con el usuario `student` en la máquina `workstation` y use el comando `lab` para calificar su trabajo. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

```
[student@workstation ~]$ lab todo-migrate grade
```

Finalizar

En `workstation`, ejecute el comando `lab todo-migrate finish` para terminar este ejercicio. Este paso es importante para garantizar que los recursos de ejercicios anteriores no impacten en los siguientes.

```
[student@workstation ~]$ lab todo-migrate finish
```

Esto concluye el trabajo de laboratorio.

Resumen

En este capítulo, aprendió lo siguiente:

- Un nombre de servicio se vuelve un nombre de host DNS para todos los pods dentro de un clúster OpenShift.
- Se crea un servicio externo con el comando `oc create service externalname`, usando la opción `external-name`.
- Red Hat recomienda que las implementaciones de producción definan los sondeos de estado.
- Red Hat ofrece un conjunto de imágenes de contenedor de middleware para implementar aplicaciones en OpenShift, incluidas aplicaciones empaquetadas como archivos fat JAR.
- El complemento (plug-in) Fabric8 Maven ofrece funciones para generar recursos de OpenShift y desencadenar procesos de OpenShift, como compilaciones e implementaciones.

Revisión completa: Red Hat OpenShift Development II: Containerizing Applications

Meta

Revisar tareas de *Red Hat OpenShift Development II: Containerizing Applications*.

Objetivos

Revisar tareas de *Red Hat OpenShift Development II: Containerizing Applications*.

Secciones

Revisión exhaustiva

Trabajo de laboratorio

- Trabajo de laboratorio: Diseño de una imagen de contenedor para OpenShift
- Trabajo de laboratorio: Contenedorización e implementación de un servicio
- Trabajo de laboratorio: Compilación e implementación de una aplicación con varios contenedores

Revisión exhaustiva

Objetivos

Tras finalizar esta sección, deberá ser capaz de revisar y actualizar las habilidades y los conocimientos aprendidos en *Red Hat OpenShift Development II: Containerizing Applications*.

Revisión de Red Hat OpenShift Development II: Containerizing Applications

Antes de comenzar la revisión integral de este curso, los estudiantes deben sentirse cómodos con los temas que se abordaron en cada capítulo.

Los estudiantes pueden consultar las secciones anteriores en el libro de texto para lecturas complementarias.

Capítulo 1, Implementación y administración de aplicaciones en un clúster OpenShift

Implementar aplicaciones mediante diferentes métodos de empaquetado de aplicaciones en un clúster OpenShift y administrar sus recursos.

- Describir la arquitectura y las nuevas características de OpenShift 4.
- Implementar una aplicación en el clúster desde un Dockerfile con la CLI.
- Implementar una aplicación desde una imagen de contenedor y administrar sus recursos mediante la consola web.
- Implementar una aplicación desde el código fuente y administrar sus recursos mediante la interfaz de línea de comandos.

Capítulo 2, Diseño de aplicaciones contenerizadas para OpenShift

Seleccionar un método de organización de aplicaciones contenedorizadas para una aplicación y empaquetarla para que se ejecute en un clúster OpenShift.

- Seleccionar un método adecuado de organización de aplicaciones contenedorizadas.
- Compilar una imagen de contenedor con directivas avanzadas de Dockerfile.
- Seleccionar un método para insertar datos de configuración en una aplicación y crear los recursos necesarios para hacerlo.

Capítulo 3, Publicación de imágenes de contenedor empresariales

Interactuar con un registro empresarial y publicar imágenes de contenedor en él.

- Gestionar las imágenes de contenedor en los registros mediante las herramientas de contenedor de Linux.

- Acceder al registro interno de OpenShift mediante las herramientas de contenedor de Linux.
- Crear secuencias de imágenes para imágenes de contenedor en registros externos.

Capítulo 4, Compilación de aplicaciones

Describir el proceso de compilación de OpenShift, desencadenar y administrar compilaciones.

- Describir el proceso de compilación de OpenShift.
- Administrar las compilaciones de la aplicación mediante el recurso BuildConfig y los comandos de CLI.
- Desencadenar el proceso de compilación con métodos soportados.
- Procesar lógicas posteriores a la compilación con un hook de compilación post-commit (posterior a la confirmación).

Capítulo 5, Personalización de compilaciones de fuente a imagen

Personalizar una imagen de compilador S2I existente y crear una nueva.

- Describir los pasos requeridos y opcionales en el proceso de compilación de fuente a imagen.
- Personalizar una imagen de compilador S2I existente con scripts.
- Crear una nueva imagen de compilador S2I con herramientas de S2I.

Capítulo 6, Creación de aplicaciones desde plantillas de OpenShift

Describir los elementos de una plantilla y crear una plantilla de aplicación con varios contenedores.

- Describir los elementos de una plantilla de OpenShift.
- Compilar una aplicación con varios contenedores desde una plantilla personalizada.

Capítulo 7, Administración de las implementaciones de aplicaciones

Monitorear el estado de la aplicación y usar diferentes métodos de implementación para aplicaciones nativas de la nube.

- Implementar sondeos de disponibilidad y ejecución.
- Seleccionar la estrategia de implementación adecuada para una aplicación nativa de la nube.
- Administrar la implementación de una aplicación con comandos de CLI.

Capítulo 8, Implementación de tuberías de integración continua y de implementación continua en OpenShift

Crear e implementar tuberías de Jenkins para facilitar la integración y el despliegue continuos con OpenShift.

Los objetivos de este capítulo no se incluyen en el trabajo de laboratorio de revisión completa.

Capítulo 9, Compilación de aplicaciones para OpenShift

Crear e implementar aplicaciones en OpenShift.

- Integrar una aplicación contenedorizada con servicios no contenedorizados.
- Implementar aplicaciones de terceros contenedorizadas siguiendo las prácticas recomendadas para OpenShift.
- Usar un tiempo de ejecución de aplicación de Red Hat OpenShift para implementar una aplicación.

► Trabajo de laboratorio

Diseño de una imagen de contenedor para OpenShift

En este trabajo de laboratorio, optimizará la imagen de contenedor para el front-end de la aplicación To Do List y lo publicará en un servidor del registro para ser consumido por un clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Optimizar un Dockerfile para reducir la cantidad de capas en la imagen de contenedor generada.
- Cambiar un Dockerfile para que la imagen de contenedor generada se pueda implementar mediante las políticas de seguridad estándares de OpenShift.
- Publicar la imagen de contenedor generada en un registro externo.
- Crear un flujo de imágenes de OpenShift, en un proyecto compartido, para implementar aplicaciones mediante la imagen de contenedor generada.

Antes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Las herramientas de contenedor de Linux.
- La imagen de contenedor de la imagen base universal (UBI) 8.
- Una bifurcación personal de GitHub y un clon local del repositorio D0288-apps, que contiene una carpeta con el origen de front-end de la aplicación To Do List y su Dockerfile en la carpeta `todo-frontend`.
- Una cuenta personal y gratuita en Quay.io.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos de la solución.

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

Instrucciones

Todos los trabajos de laboratorio de revisión de este curso se basan en el mismo caso de uso; es decir, una versión de varios contenedores de la aplicación To Do List. Cuando haya terminado todos los trabajos de laboratorio de revisión, la aplicación debe implementarse como tres pods:

- Un pod de front-end web, basado en Nginx.
- Un pod de back-end de la API HTTP, basado en Node.js.

- Un pod de base de datos, basado en MySQL.

El primer trabajo de laboratorio de revisión compila el front-end de la aplicación To Do List como imagen de contenedor personalizada mediante un Dockerfile. El equipo de desarrollo de front-end aún no ha migrado a OpenShift, y su trabajo es asegurarse de que la imagen de contenedor siga las recomendaciones de Red Hat para la implementación en OpenShift.

Debe implementar el front-end de To Do List según las siguientes especificaciones:

- Recupere las fuentes HTML de front-end de To Do List y Dockerfile desde un clon local de su bifurcación personal del repositorio Git D0288-apps, en la carpeta `todo-frontend`. Cree una bifurcación denominada `review-dockerfile` para guardar los cambios.
- El Dockerfile proporcionado genera una imagen que es compatible con los motores de contenedor de Open Container Initiative (OCI), pero puede requerir cambios para cumplir con las recomendaciones de Red Hat para OpenShift. No realice ningún cambio en las fuentes HTML y JavaScript desde el front-end de To Do List.
- Minimice la cantidad de capas en la imagen de contenedor del front-end de To Do List.
- Publique la imagen de contenedor front-end en su cuenta personal de Quay.io como `quay.io/yourquayuser/front-end:latest`.
- Cree un flujo de imágenes denominado `todo-frontend`, en el proyecto `youruser-review-common`, que apunte a la imagen de contenedor de front-end en su cuenta personal de Quay.io.
- No realice cambios a las restricciones del contexto de seguridad y las cuentas de servicios en el proyecto `youruser-review-dockerfile`. Es posible que deba conceder permisos en el proyecto `youruser-review-common` para que otros proyectos puedan acceder a sus flujos de imágenes e imágenes.
- Implemente el front-end de To Do List en el proyecto `youruser-review-dockerfile`, con `frontend` como nombre de sus recursos de OpenShift.
- Establezca la variable de entorno `BACKEND_HOST` en la implementación front-end en `api.example.com` como marcador de posición, hasta que obtenga un back-end que funcione para probar.
- Pruebe el front-end de To Do List con el nombre de host predeterminado que OpenShift genera para las nuevas rutas:
`http://frontend-youruser-review-dockerfile.apps.cluster.example.com`

Evaluación

Como usuario `student` en `workstation`, ejecute el siguiente comando para confirmar que ha realizado correctamente este ejercicio. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

No realice ningún tipo de limpieza. Los proyectos y recursos de OpenShift creados durante este trabajo de laboratorio de revisión se usarán para completar el tercer trabajo de laboratorio de revisión.

Ejecute el comando **finish** para indicar que ha completado este ejercicio:

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

Para reiniciar este trabajo de laboratorio de revisión, use el comando **cleanup**. Tenga en cuenta que no puede realizar el tercer trabajo de laboratorio de revisión si limpia este.

Esto concluye el trabajo de laboratorio.

► Solución

Diseño de una imagen de contenedor para OpenShift

En este trabajo de laboratorio, optimizará la imagen de contenedor para el front-end de la aplicación To Do List y lo publicará en un servidor del registro para ser consumido por un clúster OpenShift.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Optimizar un Dockerfile para reducir la cantidad de capas en la imagen de contenedor generada.
- Cambiar un Dockerfile para que la imagen de contenedor generada se pueda implementar mediante las políticas de seguridad estándares de OpenShift.
- Publicar la imagen de contenedor generada en un registro externo.
- Crear un flujo de imágenes de OpenShift, en un proyecto compartido, para implementar aplicaciones mediante la imagen de contenedor generada.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- Las herramientas de contenedor de Linux.
- La imagen de contenedor de la imagen base universal (UBI) 8.
- Una bifurcación personal de GitHub y un clon local del repositorio DO288-apps, que contiene una carpeta con el origen de front-end de la aplicación To Do List y su Dockerfile en la carpeta todo-frontend.
- Una cuenta personal y gratuita en Quay.io.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos de la solución.

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

Instrucciones

Todos los trabajos de laboratorio de revisión de este curso se basan en el mismo caso de uso; es decir, una versión de varios contenedores de la aplicación To Do List. Cuando haya terminado todos los trabajos de laboratorio de revisión, la aplicación debe implementarse como tres pods:

- Un pod de front-end web, basado en Nginx.
- Un pod de back-end de la API HTTP, basado en Node.js.
- Un pod de base de datos, basado en MySQL.

El primer trabajo de laboratorio de revisión compila el front-end de la aplicación To Do List como imagen de contenedor personalizada mediante un Dockerfile. El equipo de desarrollo de front-end aún no ha migrado a OpenShift, y su trabajo es asegurarse de que la imagen de contenedor siga las recomendaciones de Red Hat para la implementación en OpenShift.

Debe implementar el front-end de To Do List según las siguientes especificaciones:

- Recupere las fuentes HTML de front-end de To Do List y Dockerfile desde un clon local de su bifurcación personal del repositorio Git D0288-apps, en la carpeta todo-frontend. Cree una bifurcación denominada **review-dockerfile** para guardar los cambios.
- El Dockerfile proporcionado genera una imagen que es compatible con los motores de contenedor de Open Container Initiative (OCI), pero puede requerir cambios para cumplir con las recomendaciones de Red Hat para OpenShift. No realice ningún cambio en las fuentes HTML y JavaScript desde el front-end de To Do List.
- Minimice la cantidad de capas en la imagen de contenedor del front-end de To Do List.
- Publique la imagen de contenedor front-end en su cuenta personal de Quay.io como `quay.io/yourquayuser/front-end:latest`.
- Cree un flujo de imágenes denominado **todo-frontend**, en el proyecto **youruser-review-common**, que apunte a la imagen de contenedor de front-end en su cuenta personal de Quay.io.
- No realice cambios a las restricciones del contexto de seguridad y las cuentas de servicios en el proyecto **youruser-review-dockerfile**. Es posible que deba conceder permisos en el proyecto **youruser-review-common** para que otros proyectos puedan acceder a sus flujos de imágenes e imágenes.
- Implemente el front-end de To Do List en el proyecto **youruser-review-dockerfile**, con **frontend** como nombre de sus recursos de OpenShift.
- Establezca la variable de entorno **BACKEND_HOST** en la implementación front-end en `api.example.com` como marcador de posición, hasta que obtenga un back-end que funcione para probar.
- Pruebe el front-end de To Do List con el nombre de host predeterminado que OpenShift genera para las nuevas rutas:

`http://frontend-youruser-review-dockerfile.apps.cluster.example.com`

1. Cree una bifurcación para almacenar los cambios en el Dockerfile del front-end de To Do List y cambie el Dockerfile para seguir las recomendaciones de Red Hat para las imágenes de contenedor.
 - 1.1. Ingrese su clon local del repositorio Git D0288-apps y extraiga la bifurcación maestra del repositorio del curso para asegurarse de iniciar este ejercicio desde un estado bueno conocido:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 1.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b review-dockerfile
Switched to a new branch 'review-dockerfile'
[student@workstation D0288-apps]$ git push -u origin review-dockerfile
...output omitted...
* [new branch]      review-dockerfile -> review-dockerfile
Branch review-dockerfile set up to track remote branch review-dockerfile from
origin.
```

- 1.3. Optimice el Dockerfile provisto para reducir la cantidad de capas en la imagen de contenedor.

Abra el archivo ~/D0288-apps/todo-frontend/Dockerfile en un editor de texto y fusione las instrucciones LABEL y RUN adyacentes. Después de realizar todas las ediciones, el contenido del archivo debe ser similar a lo siguiente:

```
FROM registry.access.redhat.com/ubi8:8.0

LABEL version="1.0" \
      description="To Do List application front-end" \
      creationDate="2017-12-25" \
      updatedDate="2019-08-01"

ENV BACKEND_HOST=localhost:8081

RUN yum install -y --disableplugin=subscription-manager --nodoscs \
    nginx nginx-mod-http-perl \
    && yum clean all

COPY nginx.conf /etc/nginx/

RUN touch /run/nginx.pid \
    && chgrp -R nginx /var/log/nginx /run/nginx.pid \
    && chmod -R g+rwx /var/log/nginx /run/nginx.pid

COPY src/ /usr/share/nginx/html

EXPOSE 8080

USER nginx

CMD nginx -g "daemon off;"
```

Si no desea realizar usted mismo las ediciones, copie el archivo ~/D0288/solutions/review-dockerfile/Dockerfile-optimized en ~/D0288-apps/todo-frontend/Dockerfile y continúe con el siguiente paso.

- 1.4. Cambie el Dockerfile para generar una imagen de contenedor que se ejecute con las políticas de seguridad de OpenShift predeterminadas y que no requiera un ID de usuario fijo. Configure la imagen para ejecutarse como ID de usuario 1001, siguiendo las convenciones de OpenShift.

Abra el ~/D0288-apps/todo-frontend/Dockerfile en un editor de texto y cambie las instrucciones RUN existentes; esto cambia todos los archivos con permiso de escritura para que pertenezcan al grupo 0 (cero) y otorguen permiso de escritura al

grupo cero. Después de realizar todas las ediciones, el contenido del archivo debe ser similar a lo siguiente:

```
FROM registry.access.redhat.com/ubi8:8.0

LABEL version="1.0" \
      description="To Do List application front-end" \
      creationDate="2017-12-25" \
      updatedDate="2019-08-01"

ENV BACKEND_HOST=localhost:8081

RUN yum install -y --disableplugin=subscription-manager --nодocs \
    nginx nginx-mod-http-perl \
    && yum clean all

COPY nginx.conf /etc/nginx/

RUN touch /run/nginx.pid \
    && chgrp -R 0 /var/log/nginx /run/nginx.pid \
    && chmod -R g+rwx /var/log/nginx /run/nginx.pid

COPY src/ /usr/share/nginx/html

EXPOSE 8080

USER 1001

CMD nginx -g "daemon off;"
```

Si no desea realizar usted mismo estas ediciones, copie el archivo ~/D0288/solutions/review-dockerfile/Dockerfile-default-scc en ~/D0288-apps/todo-frontend/Dockerfile y continúe con el siguiente paso.

2. Compile la imagen de contenedor de front-end de To Do List y, de manera opcional, pruébela localmente. Una prueba local correcta no garantiza que la imagen funcione con las políticas de seguridad predeterminadas de OpenShift, pero detecta algunos errores que puede producir al editar el Dockerfile.
 - 2.1. Navegue a la carpeta ~/D0288-apps/todo-frontend y compile la imagen de contenedor todo-frontend.

```
[student@workstation D0288-apps]$ cd todo-frontend
[student@workstation todo-frontend]$ sudo podman build -t todo-frontend .
...output omitted...
STEP 19: COMMIT todo-frontend
```

- 2.2. Inicie un contenedor local para verificar que los cambios no rompieron la imagen.

```
[student@workstation todo-frontend]$ sudo podman run --name testfrontend \
> -d -p 8080:8080 todo-frontend
16a4...4a3a
```

- 2.3. Use Curl para verificar que la imagen devuelve la página de bienvenida de front-end de To Do List.

```
[student@workstation todo-frontend]$ curl -s localhost:8080 | grep h1
<h1>To Do List Application</h1>
```

- 2.4. Detenga y elimine su contenedor de prueba.

```
[student@workstation todo-frontend]$ sudo podman stop testfrontend
16a4...4a3a
[student@workstation todo-frontend]$ sudo podman rm testfrontend
16a4...4a3a
```

- 2.5. Confirme y envíe los cambios al Dockerfile.

```
[student@workstation todo-frontend]$ git commit -a -m 'Fixed for OpenShift'
...output omitted...
[student@workstation todo-frontend]$ git push
...output omitted...
```

- 2.6. Deje la carpeta del proyecto Dockerfile, ya que solo debe usar la imagen publicada para el resto de este ejercicio.

```
[student@workstation todo-frontend]$ cd ~
...output omitted...
[student@workstation ~]$
```

3. Publique la imagen del contenedor de front-end de To Do List en Quay.io. Obtenga las variables de configuración del aula de /usr/local/etc/ocp4.config antes de realizar la operación de inserción.

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

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

- 3.2. Inicie sesión en Quay.io como root, de modo que el comando Skopeo en el siguiente paso pueda crear una nueva imagen en su cuenta de Quay.io.

```
[student@workstation ~]$ sudo podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password: your Quay.io password
Login Succeeded!
```

- 3.3. Envíe la imagen local de contenedor de todo-frontend para Quay.io.

```
[student@workstation ~]$ sudo skopeo copy \
> containers-storage:localhost/todo-frontend \
> docker://quay.io/${RHT_OCP4_QUAY_USER}/todo-frontend
...output omitted...
Writing manifest to image destination
Storing signatures
```

4. Ponga la imagen de contenedor de front-end de To Do List a disposición de los usuarios de OpenShift como el flujo de imágenes todo-frontend en el proyecto `youruser-review-common`.

- 4.1. Inicie sesión en OpenShift con su cuenta de usuario desarrollador y, a continuación, cree el proyecto `youruser-review-common`.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-review-common
Now using project "youruser-review-common" on server
"https://api.cluster.domain.example.com:6443".
```

- 4.2. Inicie sesión de nuevo en Quay.io, esta vez como `student`, para guardar un token de acceso para el secreto que cree en el paso siguiente.

```
[student@workstation ~]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password: your Quay.io password
Login Succeeded!
```

- 4.3. Cree un secreto con el token de acceso de la API de registro de contenedores almacenado por Podman.

```
[student@workstation ~]$ oc create secret generic quayio \
> --from-file .dockerconfigjson=${XDG_RUNTIME_DIR}/containers/auth.json \
> --type kubernetes.io/dockerconfigjson
secret/quayio created
```

- 4.4. Cree el flujo de imágenes todo-frontend que apunte a la imagen de contenedor en su cuenta personal de Quay.io.

```
[student@workstation ~]$ oc import-image todo-frontend --confirm \
> --reference-policy local \
> --from quay.io/${RHT_OCP4_QUAY_USER}/todo-frontend
imagestream.image.openshift.io/todo-frontend imported
...output omitted...
latest
tagged from quay.io/yourquayuser/todo-frontend
prefer registry pullthrough when referencing this tag

* quay.io/yourquayuser/todo-frontend@sha256:9ca4...4651
  Less than a second ago
...output omitted...
```

5. Implemente el flujo de imágenes de front-end de To Do List en el proyecto `youruser-review-dockerfile` y verifique que funcione.

- 5.1. Cree el proyecto `youruser-review-dockerfile`.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-review-dockerfile
Now using project "youruser-review-dockerfile" on server
"https://api.cluster.domain.example.com:6443".
```

- 5.2. Conceda a las cuentas de servicio del nuevo proyecto *youruser-review-dockerfile* acceso a los flujos de imágenes del proyecto *youruser-review-common*.

```
[student@workstation ~]$ oc policy add-role-to-group \
> -n ${RHT_OCP4_DEV_USER}-review-common system:image-puller \
> system:serviceaccounts:${RHT_OCP4_DEV_USER}-review-dockerfile
clusterrole.rbac.authorization.k8s.io/system:image-puller added:
"system:serviceaccounts:youruser-review-dockerfile"
```

- 5.3. Implemente la aplicación *frontend* mediante el flujo de imágenes *todo-frontend* del proyecto *youruser-review-common*. Configurar la variable de entorno *BACKEND_HOST* en *api.example.com*.

```
[student@workstation ~]$ oc new-app --as-deployment-config --name frontend \
> -e BACKEND_HOST=api.example.com \
> -i ${RHT_OCP4_DEV_USER}-review-common/todo-frontend
...output omitted...
--> Creating resources ...
imagestreamtag.image.openshift.io "frontend:latest" created
deploymentconfig.apps.openshift.io "frontend" created
service "frontend" created
--> Success
...output omitted...
```

- 5.4. Espere a que el pod de la aplicación esté listo y en ejecución.

```
[student@workstation ~]$ oc get pod
NAME          READY   STATUS    RESTARTS   AGE
frontend-1-deploy  0/1     Completed   0          31s
frontend-1-8mwjm  1/1     Running    0          22s
```

- 5.5. Exponga el pod de la aplicación *frontend*:

```
[student@workstation ~]$ oc expose svc frontend
route "frontend" exposed
```

- 5.6. Obtenga el nombre de host de la ruta:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT   ...
frontend  frontend-youruser-review-dockerfile.apps.cluster.example.com ...
```

- 5.7. Pruebe la aplicación con el comando *curl* y el nombre de host del paso anterior.

```
[student@workstation ~]$ curl -s \  
> http://frontend-${RHT_OCP4_DEV_USER}-review-dockerfile.\ \  
> ${RHT_OCP4_WILDCARD_DOMAIN} | grep h1  
<h1>To Do List Application</h1>
```

Evaluación

Como usuario `student` en `workstation`, ejecute el siguiente comando para confirmar que ha realizado correctamente este ejercicio. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

No realice ningún tipo de limpieza. Los proyectos y recursos de OpenShift creados durante este trabajo de laboratorio de revisión se usarán para completar el tercer trabajo de laboratorio de revisión.

Ejecute el comando `finish` para indicar que ha completado este ejercicio:

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

Para reiniciar este trabajo de laboratorio de revisión, use el comando `cleanup`. Tenga en cuenta que no puede realizar el tercer trabajo de laboratorio de revisión si limpia este.

Esto concluye el trabajo de laboratorio.

► Trabajo de laboratorio

Contenerización e implementación de un servicio

En este trabajo de laboratorio, implementará el back-end de la aplicación To Do List desde el código fuente, usando la configuración y las credenciales de acceso de la base de datos externalizadas a mapas y secretos de configuración, y un script S2I personalizado.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear un secreto para almacenar las credenciales de acceso de la base de datos.
- Implementar un pod de base de datos MySQL efímera usando el secreto para brindar información de configuración.
- Personalizar la compilación S2I del back-end de To Do List agregando un script de ensamblaje personalizado.
- Implementar el back-end de To Do List del código fuente en un repositorio Git, y pasar variables de entorno de compilación.
- Crear un mapa de configuración para almacenar parámetros de configuración de la aplicación.
- Cambiar la configuración de implementación del back-end de To Do List para usar el mapa de configuración y el secreto.
- Verificar que el back-end de To Do List inicie correctamente la base de datos de MySQL.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen de contenedor de MySQL Server 5.7.
- Imagen de compilador Node.js 12.
- Una bifurcación personal de GitHub y un clon local del repositorio DO288-apps, que contiene la carpeta con el origen de front-end de la aplicación To Do List y su Dockerfile en la carpeta todo-backend.
- Acceso a un servidor Nexus con dependencias NPM requeridas por la aplicación.
- Finalización exitosa del trabajo de laboratorio de revisión integral anterior.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos de la solución.

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

Instrucciones

Todos los trabajos de laboratorio de revisión de este curso se basan en el mismo caso de uso; es decir, una versión de varios contenedores de la aplicación To Do List. Cuando haya terminado todos los trabajos de laboratorio de revisión, la aplicación debe implementarse como tres pods:

- Un pod de front-end web, basado en Nginx.
- Un pod de back-end de la API HTTP, basado en Node.js.
- Un pod de base de datos, basado en MySQL.

El segundo trabajo de laboratorio de revisión compila el back-end de To Do List usando el proceso OpenShift Source-to-Image (S2I). El equipo de desarrollo de back-end está comenzando a usar OpenShift, y su trabajo es asegurarse de que el proceso de compilación cumpla con los estándares internos de la organización. También debe asegurarse de que la implementación siga las recomendaciones de Red Hat para externalizar la configuración de aplicaciones implementadas en OpenShift.

Debe implementar el back-end de To Do List según las siguientes especificaciones:

- Implemente un servidor MySQL en el proyecto `youruser-review-service` usando `tododb` como nombre de sus recursos de OpenShift y la imagen de contenedor `rhsc1/mysql-57-rhel7` del registro público de Red Hat.
- Almacene las credenciales de acceso de la base de datos en un secreto denominado `tododb` con las claves: `user` y `password`. Use este secreto para inicializar variables de entorno para los pods de la base de datos y de back-end.
- Las variables del entorno requeridas por el pod de la base de datos son `MYSQL_USER`, `MYSQL_PASSWORD` y `MYSQL_DATABASE`.
- El nombre de usuario para acceder a la base de datos es `todoapp`, la contraseña es `mypass` y el nombre de la base de datos es `todo`.
- Recupere las fuentes Node.js de back-end de To Do List y los scripts S2I desde un clon local de su bifurcación personal del repositorio Git D0288-apps, en la carpeta `todo-backend`. Cree una bifurcación denominada `review-service` para guardar los cambios.
- Compile e implemente el back-end de To Do List en el proyecto `youruser-review-service`, con `backend` como nombre de sus recursos de OpenShift. Use la etiqueta del flujo de imágenes `nodejs:12` como el compilador S2I.
- Descargue las dependencias del módulo npm requeridas de la siguiente URL, que provee como el valor de la variable de entorno de compilación `npm_config_registry`:

`http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs`

- Las variables de entorno requeridas por el pod de la aplicación son `DATABASE_USER`, `DATABASE_PASSWORD`, `DATABASE_NAME` y `DATABASE_SVC`. Inicialícelos desde el mismo secreto `tododb` usado para inicializar las variables de entorno para el pod de la base de datos.
- Almacene los parámetros de configuración de la aplicación en un mapa de configuración denominado `todoapp` con la clave `init`.
- El único parámetro de configuración no relacionado con el acceso a la base de datos es la variable de entorno `DATABASE_INIT`. Pase el valor `true` para forzar a la aplicación a crear las tablas de la base de datos durante el inicio.

- Integre el proceso de compilación con el sistema de administración del ciclo de vida de la aplicación para la organización. Use el script `lifecycle.sh` de la carpeta `~/D0288/labs/review-service` como marcador de posición, hasta que el script de integración real esté disponible. No realice cambios al script, excepto para llamar al script de ensamblaje S2I provisto por la imagen del compilador Node.js.
- Pruebe el back-end de To Do List con el nombre de host predeterminado que OpenShift genera para las nuevas rutas:

`http://backend-youruser-review-service.apps.cluster.domain.example.com`

Use el siguiente punto de entrada de la API HTTP que devuelve el número de ítems de To Do List:

`/todo/api/items-count`

Si obtiene cero ítems, el back-end funciona y puede acceder a la base de datos.

Este trabajo de laboratorio de revisión no requiere nada del anterior y sus recursos no son usados por el tercer trabajo de laboratorio de revisión como entrada para crear la plantilla para implementar la aplicación To Do List completa. En el tercer trabajo de laboratorio de revisión, se le proporcionará una plantilla limpia basada en los recursos de este trabajo, ya sea que complete este segundo trabajo de revisión o no.

Evaluación

Como usuario `student` en `workstation`, ejecute el siguiente comando para confirmar que ha realizado correctamente este ejercicio. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

No realice ningún tipo de limpieza. Puede continuar con el tercer trabajo de laboratorio de revisión sin completar este segundo trabajo; sin embargo, debe completar el primer trabajo de laboratorio de revisión antes de iniciar el tercero.

Ejecute el comando `finish` para indicar que ha completado este ejercicio:

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

Para reiniciar este trabajo de laboratorio de revisión, use el comando `cleanup`. Tenga en cuenta que no puede realizar el tercer trabajo de laboratorio de revisión si limpia este.

Esto concluye el trabajo de laboratorio.

► Solución

Contenerización e implementación de un servicio

En este trabajo de laboratorio, implementará el back-end de la aplicación To Do List desde el código fuente, usando la configuración y las credenciales de acceso de la base de datos externalizadas a mapas y secretos de configuración, y un script S2I personalizado.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Crear un secreto para almacenar las credenciales de acceso de la base de datos.
- Implementar un pod de base de datos MySQL efímera usando el secreto para brindar información de configuración.
- Personalizar la compilación S2I del back-end de To Do List agregando un script de ensamblaje personalizado.
- Implementar el back-end de To Do List del código fuente en un repositorio Git, y pasar variables de entorno de compilación.
- Crear un mapa de configuración para almacenar parámetros de configuración de la aplicación.
- Cambiar la configuración de implementación del back-end de To Do List para usar el mapa de configuración y el secreto.
- Verificar que el back-end de To Do List inicie correctamente la base de datos de MySQL.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen de contenedor de MySQL Server 5.7.
- Imagen de compilador Node.js 12.
- Una bifurcación personal de GitHub y un clon local del repositorio **D0288 - apps**, que contiene la carpeta con el origen de front-end de la aplicación To Do List y su Dockerfile en la carpeta **todo-backend**.
- Acceso a un servidor Nexus con dependencias NPM requeridas por la aplicación.
- Finalización exitosa del trabajo de laboratorio de revisión integral anterior.

Ejecute el siguiente comando en **workstation** para validar los requisitos previos y descargar los archivos de la solución.

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

Instrucciones

Todos los trabajos de laboratorio de revisión de este curso se basan en el mismo caso de uso; es decir, una versión de varios contenedores de la aplicación To Do List. Cuando haya terminado todos los trabajos de laboratorio de revisión, la aplicación debe implementarse como tres pods:

- Un pod de front-end web, basado en Nginx.
- Un pod de back-end de la API HTTP, basado en Node.js.
- Un pod de base de datos, basado en MySQL.

El segundo trabajo de laboratorio de revisión compila el back-end de To Do List usando el proceso OpenShift Source-to-Image (S2I). El equipo de desarrollo de back-end está comenzando a usar OpenShift, y su trabajo es asegurarse de que el proceso de compilación cumpla con los estándares internos de la organización. También debe asegurarse de que la implementación siga las recomendaciones de Red Hat para externalizar la configuración de aplicaciones implementadas en OpenShift.

Debe implementar el back-end de To Do List según las siguientes especificaciones:

- Implemente un servidor MySQL en el proyecto *youruser-review-service* usando *tododb* como nombre de sus recursos de OpenShift y la imagen de contenedor *rhsc1/mysql-57-rhel7* del registro público de Red Hat.
- Almacene las credenciales de acceso de la base de datos en un secreto denominado *tododb* con las claves: *user* y *password*. Use este secreto para inicializar variables de entorno para los pods de la base de datos y de back-end.
- Las variables del entorno requeridas por el pod de la base de datos son *MYSQL_USER*, *MYSQL_PASSWORD* y *MYSQL_DATABASE*.
- El nombre de usuario para acceder a la base de datos es *todoapp*, la contraseña es *mypass* y el nombre de la base de datos es *todo*.
- Recupere las fuentes Node.js de back-end de To Do List y los scripts S2I desde un clon local de su bifurcación personal del repositorio Git *D0288-apps*, en la carpeta *todo-backend*. Cree una bifurcación denominada *review-service* para guardar los cambios.
- Compile e implemente el back-end de To Do List en el proyecto *youruser-review-service*, con *backend* como nombre de sus recursos de OpenShift. Use la etiqueta del flujo de imágenes *nodejs:12* como el compilador S2I.
- Descargue las dependencias del módulo npm requeridas de la siguiente URL, que provee como el valor de la variable de entorno de compilación *npm_config_registry*:

```
http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs
```

- Las variables de entorno requeridas por el pod de la aplicación son *DATABASE_USER*, *DATABASE_PASSWORD*, *DATABASE_NAME* y *DATABASE_SVC*. Inicialícelos desde el mismo secreto *tododb* usado para inicializar las variables de entorno para el pod de la base de datos.
- Almacene los parámetros de configuración de la aplicación en un mapa de configuración denominado *todoapp* con la clave *init*.
- El único parámetro de configuración no relacionado con el acceso a la base de datos es la variable de entorno *DATABASE_INIT*. Pase el valor *true* para forzar a la aplicación a crear las tablas de la base de datos durante el inicio.

- Integre el proceso de compilación con el sistema de administración del ciclo de vida de la aplicación para la organización. Use el script `lifecycle.sh` de la carpeta `~/D0288/labs/review-service` como marcador de posición, hasta que el script de integración real esté disponible. No realice cambios al script, excepto para llamar al script de ensamblaje S2I provisto por la imagen del compilador Node.js.
- Pruebe el back-end de To Do List con el nombre de host predeterminado que OpenShift genera para las nuevas rutas:

```
http://backend-youruser-review-service.apps.cluster.domain.example.com
```

Use el siguiente punto de entrada de la API HTTP que devuelve el número de ítems de To Do List:

```
/todo/api/items-count
```

Si obtiene cero ítems, el back-end funciona y puede acceder a la base de datos.

Este trabajo de laboratorio de revisión no requiere nada del anterior y sus recursos no son usados por el tercer trabajo de laboratorio de revisión como entrada para crear la plantilla para implementar la aplicación To Do List completa. En el tercer trabajo de laboratorio de revisión, se le proporcionará una plantilla limpia basada en los recursos de este trabajo, ya sea que complete este segundo trabajo de revisión o no.

1. Obtenga las variables de configuración del aula de `/usr/local/etc/ocp4.config`, inicie sesión en OpenShift y cree el proyecto `youruser-review-service`. Implemente un servidor MySQL al proyecto. Cuando la implementación de la base de datos haya finalizado, desactive los desencadenadores de cambio de configuración.

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

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

- 1.2. Inicie sesión en OpenShift con su cuenta de usuario desarrollador y cree el proyecto `youruser-review-service`.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-review-service
Now using project "youruser-review-service" on server
"https://api.cluster.domain.example.com:6443".
```

- 1.3. Implemente un servidor de base de datos efímera de la imagen de contenedor MySQL. Proporcione valores para todas las variables de entorno requeridas.

```
[student@workstation ~]$ oc new-app --as-deployment-config --name tododb \
> --docker-image registry.access.redhat.com/rhscl/mysql-57-rhel7 \
> -e MYSQL_USER=todoapp \
> -e MYSQL_PASSWORD=mypass \
> -e MYSQL_DATABASE=todo
```

```
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "tododb" created
  deploymentconfig.apps.openshift.io "tododb" created
  service "tododb" created
--> Success
...output omitted...
```

- 1.4. Espere hasta que el pod de la base de datos esté listo y en ejecución.

```
[student@workstation ~]$ oc get pod
NAME        READY   STATUS    RESTARTS   AGE
tododb-1-deploy  0/1     Completed  0          36s
tododb-1-tjmvs  1/1     Running   0          28s
```

- 1.5. Desactive los desencadenadores de configuración en la configuración de implementación de la base de datos. Esto no se requiere estrictamente, pero acelera los próximos pasos ya que evita varias implementaciones.

```
[student@workstation ~]$ oc set triggers dc/tododb --from-config --remove
deploymentconfig.apps.openshift.io/tododb triggers updated
```

2. Cree el secreto tododb para almacenar los parámetros de conexión de la base de datos y, luego, cambie la configuración de implementación para inicializar las variables de entorno del secreto.
- 2.1. Cree el secreto tododb con claves para almacenar el nombre de usuario y la contraseña.

```
[student@workstation ~]$ oc create secret generic tododb \
> --from-literal user=todoapp \
> --from-literal password=mypass
secret/tododb created
```

- 2.2. Cambie las variables de entorno en la configuración de implementación de la base de datos para usar las claves del secreto. Use la opción `--prefix MYSQL_` para garantizar que los nombres de la variable de entorno coincidan con los esperados por el pod de la base de datos.

```
[student@workstation ~]$ oc set env dc/tododb \
> --prefix MYSQL_ \
> --from secret/tododb
deploymentconfig.apps.openshift.io/tododb updated
```

- 2.3. Verifique que la configuración de implementación inicialice las variables de entorno del secreto.

```
[student@workstation ~]$ oc set env dc/tododb --list
# deploymentconfigs tododb, container tododb
MYSQL_DATABASE=todo
# MYSQL_PASSWORD from secret tododb, key password
# MYSQL_USER from secret tododb, key user
```

3. Vuelva a implementar la base de datos para aplicar cambios a la configuración de implementación.
 - 3.1. Implemente un nuevo pod de base de datos usando la configuración de implementación actualizada.

```
[student@workstation ~]$ oc rollout latest dc/tododb
deploymentconfig.apps.openshift.io/tododb rolled out
```

- 3.2. Espere que el nuevo pod de base de datos esté listo y en ejecución.

| NAME | READY | STATUS | RESTARTS | AGE |
|-----------------|-------|-----------|----------|-------|
| tododb-1-deploy | 0/1 | Completed | 0 | 3m45s |
| tododb-2-deploy | 0/1 | Completed | 0 | 36s |
| tododb-2-dgxwm | 1/1 | Running | 0 | 46s |

- 3.3. Verifique que las variables de entorno de la base de datos se hayan inicializado correctamente.

```
[student@workstation ~]$ oc rsh tododb-2-dgxwm env | grep MYSQL_
MYSQL_DATABASE=todo
MYSQL_PASSWORD=mypass
MYSQL_USER=todoapp
...output omitted...
```

4. Personalice la compilación de la aplicación con scripts S2I personalizados.

- 4.1. Ingrese su clon local del repositorio Git D0288-apps y compruebe la bifurcación maestra del repositorio del curso para obtener las fuentes del back-end de To Do List:

```
[student@workstation ~]$ cd D0288-apps
[student@workstation D0288-apps]$ git checkout master
...output omitted...
```

- 4.2. Cree una nueva bifurcación para guardar los cambios que realice durante este ejercicio:

```
[student@workstation D0288-apps]$ git checkout -b review-service
Switched to a new branch 'review-service'
[student@workstation D0288-apps]$ git push -u origin review-service
...output omitted...
* [new branch]      review-service -> review-service
Branch review-service set up to track remote branch review-service from origin.
```

- 4.3. Cree una carpeta para almacenar scripts S2I personalizados y, luego, copie el script `lifecycle.sh` de modo que se use como el script S2I de ensamblaje.

Copie el script `lifecycle.sh` en `./.s2i/bin/assemble` de modo que sea ejecutado por el proceso S2I OpenShift.

```
[student@workstation D0288-apps]$ mkdir -p ~/D0288-apps/todo-backend/.s2i/bin
[student@workstation D0288-apps]$ cp ~/D0288/labs/review-service/lifecycle.sh \
> ~/D0288-apps/todo-backend/.s2i/bin/assemble
```

- 4.4. Inspeccione la etiqueta del flujo de imágenes nodejs:12 para determinar la ubicación de los scripts S2I dentro de su imagen de compilador.

```
[student@workstation D0288-apps]$ oc describe istag nodejs:12 -n openshift \
> | grep io.openshift.s2i.scripts-url
"io.openshift.s2i.scripts-url": "image:///usr/libexec/s2i",
```

- 4.5. Edite el script ~/D0288-apps/todo-backend/.s2i/bin/assemble para invocar el script S2I de ensamblaje estándar de la imagen de compilador S2I Node.js.

```
#!/bin/bash

echo "Performing the S2I build..."

#TODO: add call to the standard S2I assemble script
/usr/libexec/s2i/assemble

rc=$?

if [ $rc -eq 0 ]; then
    echo "Recording successful build on the life cycle management system..."
else
    echo "Not calling the life cycle management system: S2I build failed!"
fi
exit $rc
```

- 4.6. Confirme e inserte (push) los cambios en el repositorio Git.

```
[student@workstation D0288-apps]$ cd ~/D0288-apps/todo-backend
[student@workstation todo-backend]$ git add .s2i
[student@workstation todo-backend]$ git commit -m 'Add custom assemble script'
...output omitted...
[student@workstation todo-backend]$ git push
...output omitted...
```

- 4.7. Deje la carpeta del proyecto Node.js.

```
[student@workstation todo-backend]$ cd ~
[student@workstation ~]$
```

5. Implemente el back-end de la aplicación To Do List del código fuente. No pase ningún valor para DATABASE_INIT. Pase valores únicamente para las variables de entorno que ofrecen parámetros de acceso a la base de datos. Cuando la implementación haya finalizado, desactive los desencadenadores de cambio de configuración.

- 5.1. Implemente el back-end de To Do List desde el código fuente en el repositorio Git. Proporcione valores para todas las variables de entorno requeridas. Recuerde incluir

el indicador para habilitar un recurso DeploymentConfig, el valor para la variable del entorno de compilación `npm_config_registry`, la opción `--contextDir` y el nombre de la bifurcación en la URL de Git:

```
[student@workstation ~]$ oc new-app --as-deployment-config --name backend \
> --build-env npm_config_registry=\
> http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs \
> -e DATABASE_NAME=todo \
> -e DATABASE_USER=todoapp \
> -e DATABASE_PASSWORD=mypass \
> -e DATABASE_SVC=tododb \
> --context-dir todo-backend \
> nodejs:12-https://github.com/${RHT_OCP4_GITHUB_USER}/D0288-apps#review-service
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "backend" created
buildconfig.build.openshift.io "backend" created
deploymentconfig.apps.openshift.io "backend" created
service "backend" created
--> Success
...output omitted...
```

- 5.2. Espere a que se complete la compilación. Verifique que los registros de compilación muestren el mensaje generado por el script `lifecycle.sh`.

```
[student@workstation ~]$ oc logs -f bc/backend
...output omitted...
Recording successful build on the life cycle management system...
...output omitted...
Push successful
```

- 5.3. Espere a que el pod de la aplicación esté listo y en ejecución.

```
[student@workstation ~]$ oc get pod
NAME        READY   STATUS    RESTARTS   AGE
backend-1-4k3nz  1/1     Running   0          1m
backend-1-build  0/1     Completed  0          2m
backend-1-deploy 0/1     Completed  0          2m
...output omitted...
tododb-2-dgxwm  1/1     Running   0          35m
```

- 5.4. Desactive los desencadenadores de configuración en la configuración de implementación de la aplicación. Esto no se requiere estrictamente, pero acelera los próximos pasos ya que evita varias implementaciones.

```
[student@workstation ~]$ oc set triggers dc/backend --from-config --remove
deploymentconfig.apps.openshift.io/backend triggers updated
```

- 6.** Cambie la implementación de la aplicación para inicializar las variables de entorno desde el secreto.
- 6.1. Cambie las variables de entorno en la configuración de implementación de la aplicación para usar las claves del secreto. Use la opción `--prefix DATABASE_` para garantizar

que los nombres de la variable de entorno coincidan con los esperados por el pod de la aplicación de back-end.

```
[student@workstation ~]$ oc set env dc/backend \
> --prefix DATABASE_ \
> --from secret/tododb
deploymentconfig.apps.openshift.io/backend updated
```

- 6.2. Verifique que la configuración de implementación inicialice las variables de entorno del secreto.

```
[student@workstation ~]$ oc set env dc/backend --list
# deploymentconfigs backend, container backend
DATABASE_NAME=todo
# DATABASE_PASSWORD from secret tododb, key password
DATABASE_SVC=tododb
# DATABASE_USER from secret tododb, key user
```

No fuerce una nueva implementación, ya que esto se hace en un paso posterior.

7. Cree el mapa de configuración de todoapp y cambie la implementación de back-end de To Do List para inicializar los parámetros de la aplicación desde el mapa de configuración.

- 7.1. Cree el mapa de configuración todoapp con una única clave para almacenar el parámetro de configuración DATABASE_INIT. Recuerde que el prefijo DATABASE_ no forma parte de la clave del mapa de configuración.

```
[student@workstation ~]$ oc create cm todoapp --from-literal init=true
configmap "todoapp" created
```

- 7.2. Cambie las variables de entorno en la configuración de implementación de la aplicación para usar las claves del mapa de configuración.

```
[student@workstation ~]$ oc set env dc/backend \
> --prefix=DATABASE_ \
> --from=cm/todoapp
deploymentconfig.apps.openshift.io/backend updated
```

- 7.3. Verifique que la configuración de implementación inicialice las variables de entorno DATABASE_INIT desde el mapa de configuración.

```
[student@workstation ~]$ oc set env dc/backend --list
# deploymentconfigs backend, container backend
DATABASE_NAME=todo
# DATABASE_PASSWORD from secret tododb, key password
DATABASE_SVC=tododb
# DATABASE_USER from secret tododb, key user
# DATABASE_INIT from configmap todoapp, key init
```

8. Vuelva a implementar la aplicación para aplicar cambios realizados a la configuración de implementación.

- 8.1. Implemente un nuevo pod de aplicación con la configuración de implementación actualizada.

```
[student@workstation ~]$ oc rollout latest dc/backend
deploymentconfig.apps.openshift.io/backend rolled out
```

- 8.2. Espere a que el nuevo pod de la aplicación esté listo y en ejecución.

```
[student@workstation ~]$ oc get pod
NAME          READY   STATUS    RESTARTS   AGE
backend-1-build  0/1     Completed  0          31m
backend-1-deploy 0/1     Completed  0          31m
backend-2-1bjrv  1/1     Running   0          26s
backend-2-deploy 0/1     Completed  0          31m
...output omitted...
tododb-2-dgxwm  1/1     Running   0          35m
```

- 8.3. Verifique que las variables de entorno de la aplicación se hayan inicializado correctamente.

```
[student@workstation ~]$ oc rsh backend-2-1bjrv env | grep DATABASE_
DATABASE_NAME=todo
DATABASE_PASSWORD=mypass
DATABASE_SVC=tododb
DATABASE_USER=todoapp
DATABASE_INIT=true
```

9. Use el comando `curl` para probar el back-end de To Do List. `/todo/api/items-count` debe devolver cero ítems y no debe devolver un error de base de datos. Verifique también que la base de datos contenga una única tabla vacía denominada `Items` (ítems).

- 9.1. Exponga la implementación de backend.

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

- 9.2. Obtenga el nombre de host de la ruta:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT
backend   backend-youruser-review-service.apps.cluster.domain.example.com ...
```

- 9.3. Use el comando `curl` y el nombre de host del paso anterior para verificar que el back-end de To Do List haya inicializado la base de datos.

```
[student@workstation ~]$ curl -si \
> backend-${RHT_OCP4_DEV_USER}-review-service.${RHT_OCP4_WILDCARD_DOMAIN}\ \
> /todo/api/items-count
HTTP/1.1 200 OK
...
{"count":0}
```

- 9.4. Cree un túnel de reenvío de puertos al pod de la base de datos. Este y el próximo paso no son realmente necesarios, pero podrían ser útiles si necesita solucionar problemas de diferencias de variables del entorno entre la base de datos y el pod de la aplicación.

```
[student@workstation ~]$ oc port-forward tododb-2-dgxwm 30306:3306
Forwarding from 127.0.0.1:30306 -> 3306
Forwarding from [::1]:30306 -> 3306
```

- 9.5. Abra otra ventana de terminal y use el comando `mysqlshow` para verificar que el servidor MySQL tenga una base de datos denominada `todo` y una única tabla denominada `Items` (ítems).

```
[student@workstation ~]$ mysqlshow -utodoapp -pmypass -h127.0.0.1 -P30306 todo
Database: todo
+-----+
| Tables |
+-----+
| Item   |
+-----+
```

- 9.6. Ingrese `Ctrl+C` para finalizar el túnel de reenvío de puertos.

Evaluación

Como usuario `student` en `workstation`, ejecute el siguiente comando para confirmar que ha realizado correctamente este ejercicio. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

No realice ningún tipo de limpieza. Puede continuar con el tercer trabajo de laboratorio de revisión sin completar este segundo trabajo; sin embargo, debe completar el primer trabajo de laboratorio de revisión antes de iniciar el tercero.

Ejecute el comando `finish` para indicar que ha completado este ejercicio:

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

Para reiniciar este trabajo de laboratorio de revisión, use el comando `cleanup`. Tenga en cuenta que no puede realizar el tercer trabajo de laboratorio de revisión si limpia este.

Esto concluye el trabajo de laboratorio.

► Trabajo de laboratorio

Compilación e implementación de una aplicación con varios contenedores

En este trabajo de laboratorio, creará una plantilla para implementar el front-end y el back-end de la aplicación To Do List, junto con la base de datos MySQL. Agregue parámetros para que la plantilla pueda reutilizarse; también agregue sondeos de estado a la plantilla.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Cambiar las definiciones de recursos para un secreto en el archivo de la plantilla.
- Agregar parámetros a la plantilla y hacer referencia a estos parámetros desde recursos dentro de la plantilla.
- Agregar sondeos de estado a la plantilla.
- Cree la plantilla como recurso de OpenShift.
- Implemente una aplicación desde el recurso de plantilla.
- Cargar datos de prueba en el pod de la base de datos.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen de contenedor de front-end de To Do List (`todo-frontend`) en su cuenta personal de Quay.io. Creó esta imagen y la envió a su cuenta personal de Quay.io durante el primer trabajo de laboratorio de revisión integral.
- El flujo de imágenes `frontend` en el proyecto `youruser-review-common` del primer trabajo de laboratorio de revisión integral. Si no ha completado el primer trabajo de laboratorio de revisión integral, vuelva y siga los pasos de solución. Los pasos de solución proporcionan archivos y scripts terminados, para que pueda completar el trabajo de laboratorio de revisión integral más rápidamente.
- La imagen de contenedor de MySQL Server 5.7.
- Imagen de compilador Node.js 12.
- Una bifurcación personal de GitHub y un clon local del repositorio `D0288-apps`, que contiene la carpeta con el origen de front-end de la aplicación To Do List y su Dockerfile en la carpeta `todo-backend`.
- Acceso a un servidor Nexus con dependencias NPM, según lo requiera la aplicación.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos de la solución.

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

Instrucciones

Todos los trabajos de laboratorio de revisión del curso se basan en el mismo caso de uso; es decir, una versión de varios contenedores de la aplicación To Do List. Cuando haya terminado todos los trabajos de laboratorio de revisión, la aplicación se implementará como tres pods:

- Un pod de front-end web, basado en Nginx.
- Un pod de back-end de la API HTTP, basado en Node.js.
- Un pod de base de datos, basado en MySQL.

El tercer trabajo de laboratorio de revisión crea una plantilla para implementar la aplicación To Do List completa mediante definiciones de recursos creados durante los dos trabajos de laboratorio de revisión anteriores. Para ahorrar tiempo, se le proporcionan los recursos ya exportados a un archivo YAML y limpiados. Su trabajo es lograr que la plantilla pueda volver a usarse e implementar las prácticas recomendadas de Red Hat, mediante la adición de sondeos de estado a la plantilla.

Debe implementar la aplicación To Do List completa según las siguientes especificaciones:

- Cree el recurso de la plantilla `todoapp` en el proyecto `youruser-review-common`.
- La plantilla toma los siguientes parámetros. Todos estos parámetros son necesarios:
 - `DATABASE_IMAGE`: la URL de la imagen de contenedor del servidor MySQL. Su valor predeterminado es el siguiente:
`registry.access.redhat.com/rhscl/mysql-57-rhel7`.
 - `BACK-END_REPO`: la URL de las fuentes de back-end. Su valor predeterminado es el siguiente:
`https://github.com/yourgituser/D0288-apps`.
 - `BACK-END_CTXDIR`: la carpeta que contiene las fuentes de back-end. El valor predeterminado es `todo-backend`.
 - `BACK-END_BRANCH`: la bifurcación para obtener las fuentes de back-end. El valor predeterminado es `master`.
 - `NPM_PROXY`: la URL del servidor del repositorio npm. Su valor predeterminado es el siguiente:
`http://nexus-common.apps.cluster.domain.example.com/repository/nodejs`
 - `SECRET`: el secreto para hooks web de OpenShift. Su valor predeterminado se genera de manera aleatoria.
 - `PASSWORD`: la contraseña de conexión de la base de datos. No tiene valor predeterminado.
 - `HOSTNAME`: el nombre del host usado para acceder al front-end de To Do List desde un servidor web. No tiene valor predeterminado.

- **BACKEND**: el nombre de host usado por el front-end de To Do List para acceder al back-end de To Do List. No tiene valor predeterminado.
- **CLEAN_DATABASE**: indica si la aplicación inicializa la base de datos durante el inicio. Su valor predeterminado es el siguiente "false". Se requieren comillas dobles.
- Use la plantilla estándar `nodejs-mongodb-example` como referencia para que la sintaxis defina parámetros y haga referencia a parámetros dentro de la lista de recursos de la plantilla.
- El nombre de usuario de la base de datos `todoapp` se fija en la plantilla, así como el nombre de la base de datos `todo`.
- Use los siguientes puntos de entrada de la API HTTP del back-end de To Do List como sondeos de estado:
 - Disponibilidad (Readiness): `/todo/api/host`
 - Ejecución (Liveness): `/todo/api/items-count`
- El sondeo de estado de ejecución falla hasta que se complete la base de datos.
- Configure ambos sondeos con los siguientes atributos:
 - `initialDelaySeconds: 10`
 - `timeoutSeconds: 3`
- Implemente la plantilla en el proyecto `youruser-review-multicontainer`. Envíe parámetros a la plantilla para obtener los siguientes resultados:
 - La aplicación de front-end está disponible en:
`http://todoui-youruser.apps.cluster.domain.example.com`
 - La aplicación de back-end está disponible en:
`http://todoapi-youruser.apps.cluster.domain.example.com`
 - En las dos URL anteriores, use la variable de shell `RHT_OCP4_WILDCARD_DOMAIN` para proporcionar el valor de `apps.cluster.domain.example.com` y la variable de shell `RHT_OCP4_DEV_USER` para proporcionar el valor de `youruser`. Estas variables se definen en el archivo de shell `/usr/local/etc/ocp4.config`.
 - La contraseña de la base de datos es `redhat`.
 - La aplicación de back-end *no* inicializa la base de datos.
- Complete la base de datos usando el script `todo.sql` en la carpeta `~/D0288/labs/review-multicontainer`.
- Dado que la plantilla implementa el front-end de To Do List desde el flujo de imágenes que creó durante el primer trabajo de laboratorio de revisión integral, debe conceder permisos a las cuentas de servicio desde el proyecto `youruser-review-multicontainer`. No realice ningún cambio en el archivo de la plantilla, excepto para usar los parámetros, como se indicó anteriormente.

El archivo de la plantilla de inicio `todoapp.yml` se exporta desde los proyectos `youruser-review-dockerfile` y `youruser-review-service`, creados durante los trabajos de

laboratorios de revisión anteriores. Aunque no trabajará más con estos proyectos, este trabajo de laboratorio requiere el proyecto `youruser-review-common` y la imagen de contenedor y el flujo de imágenes, creados al realizar el primer trabajo de laboratorio de revisión.

Evaluación

Como usuario `student` en `workstation`, ejecute el siguiente comando para confirmar que ha realizado correctamente este ejercicio. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

Ejecute el comando `finish` para indicar que ha completado este ejercicio:

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

Para reiniciar todos los trabajos de laboratorio de revisión integral, use el comando `cleanup` para cada trabajo de laboratorio, en orden invertido.

```
[student@workstation ~]$ lab review-multicontainer cleanup
[student@workstation ~]$ lab review-service cleanup
[student@workstation ~]$ lab review-dockerfile cleanup
```

Con esto concluye la revisión exhaustiva.

► Solución

Compilación e implementación de una aplicación con varios contenedores

En este trabajo de laboratorio, creará una plantilla para implementar el front-end y el back-end de la aplicación To Do List, junto con la base de datos MySQL. Agregue parámetros para que la plantilla pueda reutilizarse; también agregue sondeos de estado a la plantilla.

Resultados

Usted deberá ser capaz de realizar lo siguiente:

- Cambiar las definiciones de recursos para un secreto en el archivo de la plantilla.
- Agregar parámetros a la plantilla y hacer referencia a estos parámetros desde recursos dentro de la plantilla.
- Agregar sondeos de estado a la plantilla.
- Cree la plantilla como recurso de OpenShift.
- Implemente una aplicación desde el recurso de plantilla.
- Cargar datos de prueba en el pod de la base de datos.

Andes De Comenzar

Para realizar este ejercicio, asegúrese de tener acceso a lo siguiente:

- Un clúster OpenShift en ejecución.
- La imagen de contenedor de front-end de To Do List (`todo-frontend`) en su cuenta personal de Quay.io. Creó esta imagen y la envió a su cuenta personal de Quay.io durante el primer trabajo de laboratorio de revisión integral.
- El flujo de imágenes `frontend` en el proyecto `youruser-review-common` del primer trabajo de laboratorio de revisión integral. Si no ha completado el primer trabajo de laboratorio de revisión integral, vuelva y siga los pasos de solución. Los pasos de solución proporcionan archivos y scripts terminados, para que pueda completar el trabajo de laboratorio de revisión integral más rápidamente.
- La imagen de contenedor de MySQL Server 5.7.
- Imagen de compilador Node.js 12.
- Una bifurcación personal de GitHub y un clon local del repositorio `D0288-apps`, que contiene la carpeta con el origen de front-end de la aplicación To Do List y su Dockerfile en la carpeta `todo-backend`.
- Acceso a un servidor Nexus con dependencias NPM, según lo requiera la aplicación.

Ejecute el siguiente comando en `workstation` para validar los requisitos previos y descargar los archivos de la solución.

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

Instrucciones

Todos los trabajos de laboratorio de revisión del curso se basan en el mismo caso de uso; es decir, una versión de varios contenedores de la aplicación To Do List. Cuando haya terminado todos los trabajos de laboratorio de revisión, la aplicación se implementará como tres pods:

- Un pod de front-end web, basado en Nginx.
- Un pod de back-end de la API HTTP, basado en Node.js.
- Un pod de base de datos, basado en MySQL.

El tercer trabajo de laboratorio de revisión crea una plantilla para implementar la aplicación To Do List completa mediante definiciones de recursos creados durante los dos trabajos de laboratorio de revisión anteriores. Para ahorrar tiempo, se le proporcionan los recursos ya exportados a un archivo YAML y limpiados. Su trabajo es lograr que la plantilla pueda volver a usarse e implementar las prácticas recomendadas de Red Hat, mediante la adición de sondeos de estado a la plantilla.

Debe implementar la aplicación To Do List completa según las siguientes especificaciones:

- Cree el recurso de la plantilla `todoapp` en el proyecto `youruser-review-common`.
- La plantilla toma los siguientes parámetros. Todos estos parámetros son necesarios:
 - `DATABASE_IMAGE`: la URL de la imagen de contenedor del servidor MySQL. Su valor predeterminado es el siguiente:

`registry.access.redhat.com/rhscl/mysql-57-rhel7.`

- `BACK-END_REPO`: la URL de las fuentes de back-end. Su valor predeterminado es el siguiente:

`https://github.com/yourgituser/D0288-apps.`

- `BACK-END_CTXDIR`: la carpeta que contiene las fuentes de back-end. El valor predeterminado es `todo-backend`.

- `BACK-END_BRANCH`: la bifurcación para obtener las fuentes de back-end. El valor predeterminado es `master`.

- `NPM_PROXY`: la URL del servidor del repositorio npm. Su valor predeterminado es el siguiente:

`http://nexus-common.apps.cluster.domain.example.com/repository/nodejs`

- `SECRET`: el secreto para hooks web de OpenShift. Su valor predeterminado se genera de manera aleatoria.

- `PASSWORD`: la contraseña de conexión de la base de datos. No tiene valor predeterminado.

- `HOSTNAME`: el nombre del host usado para acceder al front-end de To Do List desde un servidor web. No tiene valor predeterminado.

- **BACKEND**: el nombre de host usado por el front-end de To Do List para acceder al back-end de To Do List. No tiene valor predeterminado.
- **CLEAN_DATABASE**: indica si la aplicación inicializa la base de datos durante el inicio. Su valor predeterminado es el siguiente "false". Se requieren comillas dobles.
- Use la plantilla estándar `nodejs-mongodb-example` como referencia para que la sintaxis defina parámetros y haga referencia a parámetros dentro de la lista de recursos de la plantilla.
- El nombre de usuario de la base de datos `todoapp` se fija en la plantilla, así como el nombre de la base de datos `todo`.
- Use los siguientes puntos de entrada de la API HTTP del back-end de To Do List como sondeos de estado:
 - Disponibilidad (Readiness): `/todo/api/host`
 - Ejecución (Liveness): `/todo/api/items-count`
- El sondeo de estado de ejecución falla hasta que se complete la base de datos.
- Configure ambos sondeos con los siguientes atributos:
 - `initialDelaySeconds: 10`
 - `timeoutSeconds: 3`
- Implemente la plantilla en el proyecto `youruser-review-multicontainer`. Envíe parámetros a la plantilla para obtener los siguientes resultados:
 - La aplicación de front-end está disponible en:
`http://todoui-youruser.apps.cluster.domain.example.com`
 - La aplicación de back-end está disponible en:
`http://todoapi-youruser.apps.cluster.domain.example.com`
 - En las dos URL anteriores, use la variable de shell `RHT_OCP4_WILDCARD_DOMAIN` para proporcionar el valor de `apps.cluster.domain.example.com` y la variable de shell `RHT_OCP4_DEV_USER` para proporcionar el valor de `youruser`. Estas variables se definen en el archivo de shell `/usr/local/etc/ocp4.config`.
 - La contraseña de la base de datos es `redhat`.
 - La aplicación de back-end *no* inicializa la base de datos.
- Complete la base de datos usando el script `todo.sql` en la carpeta `~/D0288/labs/review-multicontainer`.
- Dado que la plantilla implementa el front-end de To Do List desde el flujo de imágenes que creó durante el primer trabajo de laboratorio de revisión integral, debe conceder permisos a las cuentas de servicio desde el proyecto `youruser-review-multicontainer`. No realice ningún cambio en el archivo de la plantilla, excepto para usar los parámetros, como se indicó anteriormente.

El archivo de la plantilla de inicio `todoapp.yml` se exporta desde los proyectos `youruser-review-dockerfile` y `youruser-review-service`, creados durante los trabajos de

laboratorios de revisión anteriores. Aunque no trabajará más con estos proyectos, este trabajo de laboratorio requiere el proyecto `youruser-review-common` y la imagen de contenedor y el flujo de imágenes, creados al realizar el primer trabajo de laboratorio de revisión.

- Revise el archivo de la plantilla de inicio.

- Copie el archivo `~/D0288/labs/review-multicontainer/todoapp.yaml` a la carpeta `/home/student/`.

Use el comando `grep` para inspeccionar la plantilla y verificar la cantidad y el tipo de recursos en la lista de objetos de la plantilla. Hay dos espacios antes de `kind:` en el comando `grep`.

```
[student@workstation ~]$ cp ~/D0288/labs/review-multicontainer/todoapp.yaml \
> ~/todoapp.yaml
[student@workstation ~]$ grep '^ kind:' ~/todoapp.yaml
  kind: ImageStream
  kind: ImageStream
  kind: ConfigMap
  kind: Secret
  kind: BuildConfig
  kind: DeploymentConfig
  kind: DeploymentConfig
  kind: Service
  kind: Service
  kind: Route
  kind: DeploymentConfig
  kind: Service
  kind: Route
```

- Use el comando `oc process` para enumerar los parámetros ya definidos en la plantilla.

```
[student@workstation ~]$ oc process --parameters -f ~/todoapp.yaml
NAME      ... VALUE
DATABASE_IMAGE   ... registry.access.redhat.com/rhscl/mysql-57-rhel7
BACK_END_REPO    ... https://github.com/yourgituser/D0288-apps
BACK_END_CTXDIR ... todo-backend
BACK_END_BRANCH ... master
NPM_PROXY        ... http://nexus-common.apps.cluster.domain.example.com/
repository/nodejs
SECRET
PASSWORD
HOSTNAME
BACKEND
```

- Agregue parámetros a la plantilla y úselos en la lista de objetos de la plantilla.

Para usar parámetros con un recurso secreto, actualice la representación binaria de los datos secretos con la representación de texto.

Inspeccione la plantilla `nodejs-mongodb-example` de `openshift`, si necesita ayuda con la sintaxis de la plantilla.

- 2.1. Cargue la configuración del entorno de clase y, a continuación, inicie sesión en OpenShift con su cuenta de usuario de desarrollador

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
```

- 2.2. Inspeccione el `nodejs-mongodb-example` del proyecto `openshift`. Realice este paso en una ventana de terminal diferente para poder ver la plantilla de muestra, a medida que realiza ediciones durante los siguientes pasos. Use el archivo como referencia para la sintaxis de parámetros, secretos y demás recursos de la plantilla.

```
[student@workstation ~]$ oc get template nodejs-mongodb-example -n openshift \
> -o yaml | less
```

- 2.3. Revise y complete los parámetros definidos al final del archivo `~/todoapp.yaml`.

Agregue el parámetro `CLEAN_DATABASE` y complete la definición del parámetro `SECRET`.

En la siguiente lista, se muestran las definiciones de parámetros completas y se resaltan los cambios que se realizarán. Puede copiar y pegar todos los cambios de este y los anteriores pasos del archivo de la plantilla de soluciones en la carpeta `~/D0288/solutions/review-multicontainer`.

```
...output omitted...
parameters:
- displayName: MySQL server container image full name (with registry)
  name: DATABASE_IMAGE
  required: true
  value: registry.access.redhat.com/rhscl/mysql-57-rhel7
- displayName: To Do List back-end Git repository URL
  name: BACK_END_REPO
  required: true
  value: https://github.com/yourgituser/D0288-apps
- displayName: To Do List back-end project root folder
  name: BACK_END_CTXDIR
  required: true
  value: todo-backend
- displayName: Git branch to build the To Do List back-end
  name: BACK_END_BRANCH
  required: true
  value: todo-backend
- displayName: Npm modules repository URL
  name: NPM_PROXY
  required: true
  value: http://nexus-common.apps.cluster.domain.example.com/repository/nodejs
- displayName: Secret for webhooks
  name: SECRET
  required: true
  from: '[a-zA-Z0-9]{40}'
  generate: expression
- displayName: MySQL database password for the todoapp user
```

```

name: PASSWORD
required: true
- displayName: Host name to access the To Do List front-end web application
  name: HOSTNAME
  required: true
- displayName: Host name to access the To Do List back-end HTTP API
  name: BACKEND
  required: true
- displayName: Flag to initialize (or not) the application database
  name: CLEAN_DATABASE
  required: true
  value: "false"

```

- 2.4. Cambie el recurso secreto en el archivo de la plantilla `~/todoapp.yaml` para ser inicializado desde parámetros de la plantilla.

Primero, revise la plantilla estándar `nodejs-mongodb-example`; luego, use el comando `oc explain secret` para obtener más información acerca de los atributos de recursos secretos, si no está seguro de las ediciones que debe realizar.

Reemplace el atributo `data` por el atributo `stringData` y reemplace los valores codificados por las claves `password` y `user` con valores de texto simple. La clave `password` hace referencia al parámetro `PASSWORD` de la plantilla:

```

...output omitted...
- apiVersion: v1
  stringData:
    password: ${PASSWORD}
    user: todoapp
  kind: Secret
  metadata:
    name: tododb
  type: Opaque
...output omitted...

```

- 2.5. Reemplace los valores del atributo de los recursos restantes en el atributo de la matriz `objects` (objetos) con referencias a los parámetros de la plantilla adecuada.

Edite las definiciones de recursos para el mapa de configuración y la ruta de front-end para usar los parámetros de la plantilla adecuada.

Las definiciones de los recursos para el flujo de imágenes de la base de datos y la configuración de compilación ya hacen referencia a los parámetros de la plantilla correcta.

En la siguiente lista, se muestran todas las referencias a parámetros en la lista de objetos de plantilla y se resaltan los atributos que se va a cambiar:

```

...output omitted...
objects:
...output omitted...
- apiVersion: image.openshift.io/v1
  kind: ImageStream
...output omitted...
  name: tododb
  spec:
...output omitted...

```

```
tags:
- annotations:
  from:
    kind: DockerImage
    name: ${DATABASE_IMAGE}
...output omitted...
- apiVersion: v1
data:
  init: ${CLEAN_DATABASE}
kind: ConfigMap
metadata:
  name: todoapp
...output omitted...
- apiVersion: v1
stringData:
  password: ${PASSWORD}
  user: todoapp
kind: Secret
metadata:
  name: tododb
...output omitted...
- apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
...output omitted...
  name: backend
spec:
...output omitted...
  source:
    contextDir: ${BACK_END_CTXDIR}
    git:
      ref: ${BACK_END_BRANCH}
      uri: ${BACK_END_REPO}
...output omitted...
  strategy:
    sourceStrategy:
      env:
        - name: npm_config_registry
          value: ${NPM_PROXY}
...output omitted...
  triggers:
    - github:
        secret: ${SECRET}
        type: GitHub
    - generic:
        secret: ${SECRET}
        type: Generic
...output omitted...
- apiVersion: route.openshift.io/v1
kind: Route
metadata:
...output omitted...
  name: backend
...output omitted...
spec:
```

```

host: ${BACKEND}
...output omitted...
- apiVersion: apps.openshift.io/v1
  kind: DeploymentConfig
...output omitted...
  name: frontend
...output omitted...
  template:
...output omitted...
  spec:
    containers:
      - env:
          - name: BACKEND_HOST
            value: ${BACKEND}
        imagePullPolicy: Always
...output omitted...
- apiVersion: route.openshift.io/v1
  kind: Route
  metadata:
...output omitted...
  name: frontend
...output omitted...
  spec:
    host: ${HOSTNAME}
...output omitted...

```

3. Agregue sondeos de estado a la configuración de implementación del back-end de To Do List dentro de la plantilla.
 - 3.1. La configuración de implementación de back-end no incluye sondeos de estado, por lo que debe agregarlos a la definición de plantilla.
Revise la plantilla estándar nodejs-mongodb-example y copie los sondeos de estado de esa plantilla al archivo ~/todoapp.yaml, agregándolos al final del atributo spec.template.spec.containers de la configuración de implementación de back-end. Cambie los atributos del sondeo para que coincidan con los requisitos de la aplicación.
En la siguiente lista, se muestra la definición del sondeo. Puede copiar y pegar todos los cambios de este y los anteriores pasos del archivo de la plantilla de soluciones en la carpeta ~/D0288/solutions/review-multicontainer.

```

...output omitted...
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
...output omitted...
  name: backend
...output omitted...
  spec:
...output omitted...
  template:
...output omitted...
  spec:
    containers:
...output omitted...

```

```

terminationMessagePolicy: File
livenessProbe:
  httpGet:
    path: /todo/api/items-count
    port: 8080
  initialDelaySeconds: 10
  timeoutSeconds: 3
readinessProbe:
  httpGet:
    path: /todo/api/host
    port: 8080
  initialDelaySeconds: 10
  timeoutSeconds: 3
  dnsPolicy: ClusterFirst
...output omitted...

```

- 3.2. Use el comando `oc new-app` con la opción `--dry-run` para verificar rápidamente las ediciones finalizadas.

Dado que el comando `oc new-app` requiere un contexto de OpenShift para funcionar, ingrese el proyecto `youruser-review-common`.

La opción `--dry-run` verifica que haya pasado los valores para todos los parámetros requeridos sin un valor predeterminado y que la definición de la plantilla esté bien formada. Esta opción no verifica que las referencias a parámetros sigan la sintaxis correcta, así como tampoco determina si algún recurso en el atributo de objetos de la plantilla tiene algún problema.

Puede pasar cualquier valor a los parámetros de la plantilla, ya que no está creando ningún recurso. El comando `oc new-app --dry-run` muestra un mensaje de error si un parámetro requerido no tiene un valor, ya sea en la definición del parámetro o en el comando.

```

[student@workstation ~]$ oc project ${RHT_OCP4_DEV_USER}-review-common
Now using project "youruser-review-common" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
[student@workstation ~]$ oc new-app --dry-run -f ~/todoapp.yaml \
> -p PASSWORD=x -p HOSTNAME=y -p BACKEND=z
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "backend" created (dry run)
  imagestream.image.openshift.io "tododb" created (dry run)
  configmap "todoapp" created (dry run)
  secret "tododb" created (dry run)
  buildconfig.build.openshift.io "backend" created (dry run)
  deploymentconfig.apps.openshift.io "backend" created (dry run)
  deploymentconfig.apps.openshift.io "tododb" created (dry run)
  service "backend" created (dry run)
  service "tododb" created (dry run)
  route.route.openshift.io "backend" created (dry run)
  deploymentconfig.apps.openshift.io "frontend" created (dry run)
  service "frontend" created (dry run)
  route.route.openshift.io "frontend" created (dry run)
--> Success (dry run)
...output omitted...

```

4. Cree y revise el recurso de la plantilla en el proyecto *youruser-review-common*.

- 4.1. Cree el recurso de la plantilla dentro del proyecto *youruser-review-common*.

```
[student@workstation ~]$ oc create -f ~/todoapp.yaml
template.template.openshift.io/todoapp created
```

- 4.2. Verifique que el recurso de la plantilla esté disponible y defina los siete [MÁS] parámetros con los valores predeterminados correctos.

```
[student@workstation ~]$ oc process --parameters todoapp
NAME          ... VALUE
DATABASE_IMAGE ... registry.access.redhat.com/rhscl/mysql-57-rhel7
BACK_END_REPO  ... https://github.com/yourgituser/D0288-apps
BACK_END_CTXDIR ... todo-backend
BACK_END_BRANCH ... master
NPM_PROXY      ... http://nexus-common.apps.cluster.domain.example.com/
repository/nodejs
SECRET         ... [a-zA-Z0-9]{40}
PASSWORD
HOSTNAME
BACKEND
CLEAN_DATABASE ... false
```

5. Implemente la plantilla en el proyecto *youruser-review-multicontainer*. Verifique que el pod de front-end y el pod de la base de datos estén listos y en ejecución, pero después de algunos segundos, el pod de back-end devuelve un estado de no listo.

- 5.1. Cree el proyecto *youruser-review-multicontainer*.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-review-multicontainer
Now using project "youruser-review-multicontainer" on server
"https://api.cluster.domain.example.com:6443".
...output omitted...
```

- 5.2. Conceda a las cuentas de servicio del nuevo proyecto *youruser-review-multicontainer* acceso a los flujos de imágenes del proyecto *youruser-review-common*.

```
[student@workstation ~]$ oc policy add-role-to-group \
> -n ${RHT_OCP4_DEV_USER}-review-common system:image-puller \
> system:serviceaccounts:${RHT_OCP4_DEV_USER}-review-multicontainer
clusterrole.rbac.authorization.k8s.io/system:image-puller added:
"system:serviceaccounts:youruser-review-multicontainer"
```

- 5.3. Implemente la aplicación To Do List desde la plantilla, pasando cualquier atributo requerido u opcional para cumplir con las especificaciones de este trabajo de laboratorio de revisión. Debe proporcionar valores para los parámetros **PASSWORD**, **CLEAN_DATABASE**, **HOSTNAME** y **BACKEND**.

```
[student@workstation ~]$ oc new-app ${RHT_OCP4_DEV_USER}-review-common/todoapp \
> -p PASSWORD=redhat \
> -p CLEAN_DATABASE=false \
```

```
> -p HOSTNAME=todoui-${RHT_OCP4_DEV_USER}.${RHT_OCP4_WILDCARD_DOMAIN} \
> -p BACKEND=todoapi-${RHT_OCP4_DEV_USER}.${RHT_OCP4_WILDCARD_DOMAIN}
...output omitted...
--> Creating resources ...
    imagestream.image.openshift.io "backend" created
    imagestream.image.openshift.io "tododb" created
    configmap "todoapp" created
    secret "tododb" created
    buildconfig.build.openshift.io "backend" created
    deploymentconfig.apps.openshift.io "backend" created
    deploymentconfig.apps.openshift.io "tododb" created
    service "backend" created
    service "tododb" created
    route.route.openshift.io "backend" created
    deploymentconfig.apps.openshift.io "frontend" created
    service "frontend" created
    route.route.openshift.io "frontend" created
--> Success
...output omitted...
```

- 5.4. Espere hasta que finalice la compilación y todos los pods estén listos y en ejecución. El pod de back-end se verá en buen estado durante los primeros segundos después de su implementación.

```
[student@workstation ~]$ oc logs -f bc/backend
...output omitted...
Push successful
[student@workstation ~]$ oc get pod
NAME          READY   STATUS    RESTARTS   AGE
backend-1-build  0/1     Completed  0          2m
backend-1-deploy 0/1     Completed  0          1m
backend-1-gcscq  1/1     Running   0          5s
frontend-1-deploy 0/1     Completed  0          2m
frontend-1-8qph0  1/1     Running   0          2m
tododb-1-deploy  0/1     Completed  0          2m
tododb-1-dvcqm  1/1     Running   0          2m
```

- 5.5. Despues de algunos segundos, el pod de back-end se reinicia debido al sondeo de ejecución.

```
[student@workstation ~]$ oc get pod
NAME          READY   STATUS    RESTARTS   AGE
...output omitted...
backend-1-gcscq  0/1     Running   1          51s
...output omitted...
```

- 5.6. Obtenga el nombre de host de la ruta de back-end:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT
...  

backend  todoapi-youruser.apps.cluster.domain.example.com ...
frontend  todoui-youruser.apps.cluster.domain.example.com ...
```

- 5.7. Use Curl para comprobar que el pod de back-end no puede consultar la tabla `Items` (`Items`). Use el punto de entrada de la API HTTP del sondeo de ejecución y el nombre de host del paso anterior.

Observe que, si espera demasiado para realizar este comando, el pod de back-end pasará al estado `CrashLoopBackOff`, y ya no podrá acceder a la API HTTP. Si esto sucede, continúe con el paso siguiente.

Puede que obtenga un error de enrutador si intenta Curl al reiniciar el pod. Si ve este error, inténtelo de nuevo.

```
[student@workstation ~]$ curl -siw "\n" \
> todoapi-${RHT_OCP4_DEV_USER}.${RHT_OCP4_WILDCARD_DOMAIN}/todo/api/items-count
HTTP/1.1 500 Internal Server Error
...
>{"message":"ER_NO_SUCH_TABLE: Table 'todo.Item' doesn't exist"}
```

6. Complete la base de datos y verifique que el pod de back-end esté en buen estado. Abra un navegador web para comprobar que la aplicación funciona según lo esperado. No realice cambios en ninguno de los ítems de la base de datos, ya que esto podría afectar el script de calificación.

- 6.1. Cree un túnel de reenvío de puertos al pod de la base de datos.

```
[student@workstation ~]$ oc port-forward tododb-1-dvcqm 30306:3306
Forwarding from 127.0.0.1:30306 -> 3306
Forwarding from [::1]:30306 -> 3306
```

- 6.2. Abra otro terminal y, luego, use un cliente MySQL para ejecutar el script `~/D0288/labs/review-multicontainer/todo.sql`. Presione Ctrl+C para finalizar el túnel de reenvío.

```
[student@workstation ~]$ mysql -h127.0.0.1 -P30306 -utodoapp -predhat todo \
> < ~/D0288/labs/review-multicontainer/todo.sql
```

- 6.3. Si demora mucho tiempo en inicializar la base de datos, el pod de back-end ingresará al estado `CrashLoopBackOff`. Si esto ocurre, vuelva a implementar el pod de back-end y espere a que el nuevo pod esté listo y en ejecución.

```
[student@workstation ~]$ oc get pod
NAME          READY   STATUS      RESTARTS   AGE
...output omitted...
backend-1-gcscq   0/1     CrashLoopBackOff   6           6m
...output omitted...
[student@workstation ~]$ oc rollout latest dc/backend
deploymentconfig.apps.openshift.io/backend rolled out
[student@workstation ~]$ oc get pod
NAME          READY   STATUS      RESTARTS   AGE
...output omitted...
backend-2-1ppd1   1/1     Running    0           1m
...output omitted...
```

- 6.4. Use Curl para verificar que el back-end de la aplicación To Do List encuentre seis ítems en la base de datos. Use el mismo punto de entrada de API HTTP como sondeo de ejecución.

```
[student@workstation ~]$ curl -siw "\n" \
> todoapi-${RHT_OCP4_DEV_USER}.${RHT_OCP4_WILDCARD_DOMAIN}/todo/api/items-count
HTTP/1.1 200 OK
...
>{"count":6}
```

- 6.5. Obtenga el nombre de host de la ruta de front-end:

```
[student@workstation ~]$ oc get route
NAME      HOST/PORT
backend   todoapi-youruser.apps.cluster.domain.example.com ...
frontend  todoui-youruser.apps.cluster.domain.example.com ...
```

- 6.6. Abra un navegador web y acceda al front-end de To Do List, con el nombre de host del paso anterior. Se muestran seis ítems en la base de datos. No realice cambios a estos ítems, ya que esto podría afectar el script de calificación.

| Id | Description | Done | |
|----|------------------|-------|---|
| 1 | Populate ex... | true | X |
| 2 | Create projec... | true | X |
| 3 | Deploy To ... | true | X |
| 4 | Create servi... | false | X |
| 5 | Add endpoi... | false | X |
| 6 | Verify that t... | false | X |

Add Task

Description:

Add Description.

Completed:

Clear **Save**

Evaluación

Como usuario `student` en `workstation`, ejecute el siguiente comando para confirmar que ha realizado correctamente este ejercicio. Corrija los errores informados y vuelva a ejecutar el comando hasta obtener un resultado satisfactorio.

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

Finalizar

Ejecute el comando `finish` para indicar que ha completado este ejercicio:

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

Para reiniciar todos los trabajos de laboratorio de revisión integral, use el comando `cleanup` para cada trabajo de laboratorio, en orden invertido.

```
[student@workstation ~]$ lab review-multicontainer cleanup  
[student@workstation ~]$ lab review-service cleanup  
[student@workstation ~]$ lab review-dockerfile cleanup
```

Con esto concluye la revisión exhaustiva.

apéndice A

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).

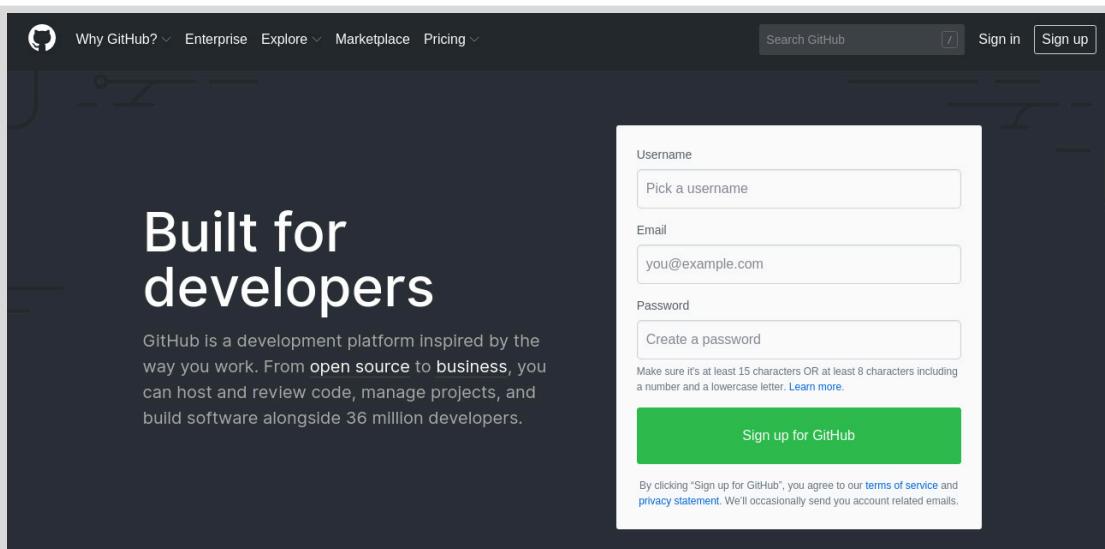


Figura A.1: Creación de una cuenta de GitHub

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. Después 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.

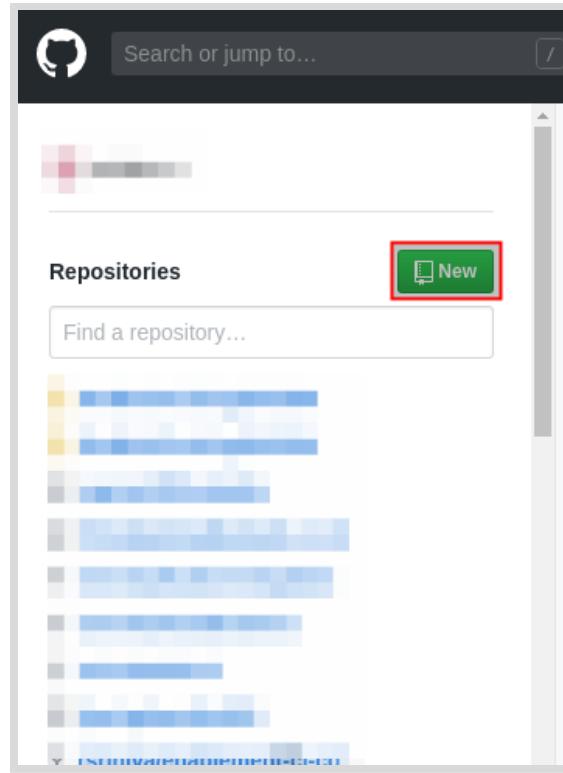


Figura A.2: Creación de un nuevo repositorio Git

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).

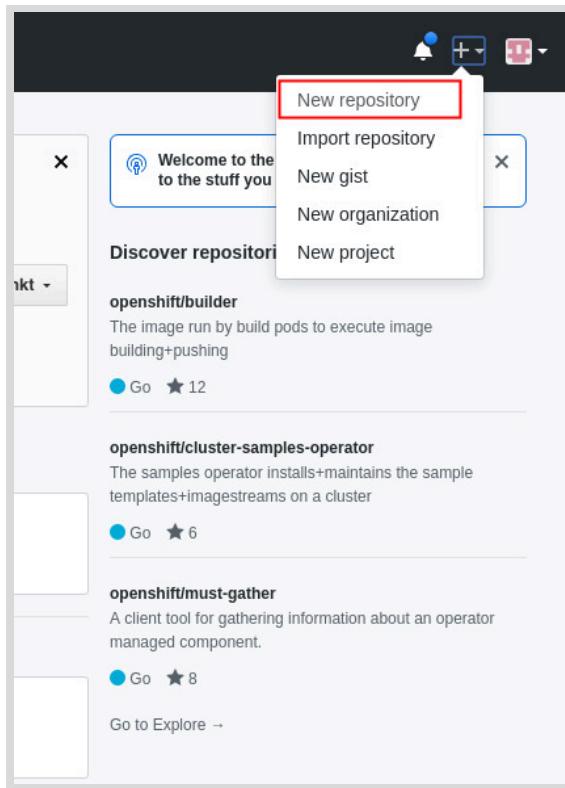


Figura A.3: Creación de un nuevo repositorio Git



Referencias

Registro para obtener una nueva cuenta de GitHub

<https://help.github.com/en/articles/signing-up-for-a-new-github-account>

apéndice B

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).

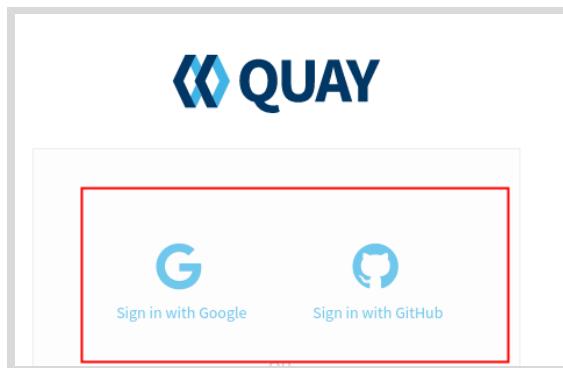


Figura B.1: Inicie sesión con las credenciales de Google o GitHub.

De manera alternativa, haga clic en **Create account** (Crear cuenta) para crear una nueva cuenta.

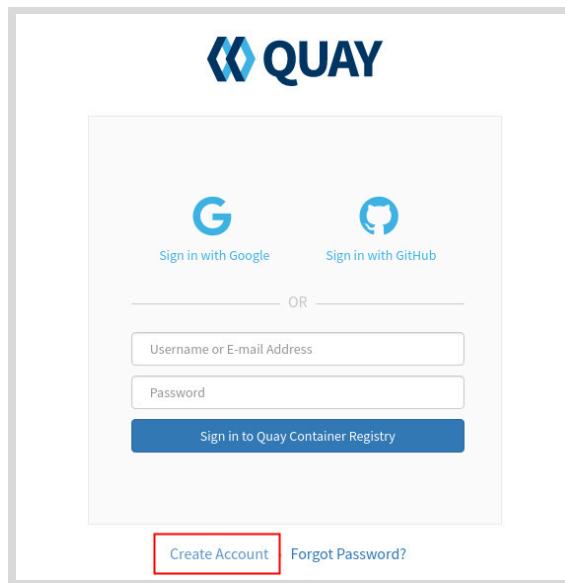


Figura B.2: Creación de una nueva cuenta

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. Despues 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).

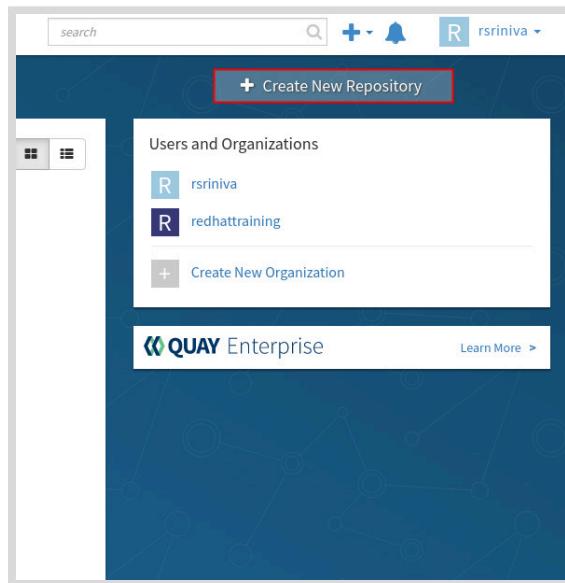


Figura B.3: Creación de un nuevo repositorio de imágenes

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).

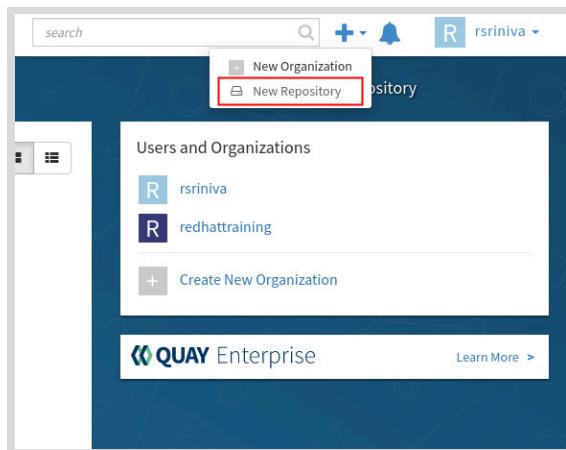


Figura B.4: Creación de un nuevo repositorio de imágenes



Referencias

Comencemos con Quay.io

<https://docs.quay.io/solution/getting-started.html>

apéndice C

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 D0288-apps y cambie a la bifurcación maestra:

```
[student@workstation ~]$ cd ~/D0288-apps
[student@workstation D0288-apps]$ git checkout master
```

- Elimine su bifurcación local:

```
[student@workstation D0288-apps]$ git branch -d your-branch
```

- Elimine la bifurcación remota en su cuenta de GitHub personal:

```
[student@workstation D0288-apps]$ git push origin --delete your-branch
```

- Use el subcomando `start` para reiniciar el ejercicio:

```
[student@workstation D0288-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:

- Ejecute el subcomando `finish` del ejercicio para limpiar todos los recursos.

```
[student@workstation ~]$ lab your-exercise finish
```

- Ingrese su clon local del repositorio D0288-apps y descarte los cambios pendientes en la bifurcación actual mediante `git stash`:

```
[student@workstation ~]$ cd ~/D0288-apps
[student@workstation D0288-apps]$ git stash
```

- Cambie a la bifurcación maestra de su repositorio local:

```
[student@workstation D0288-apps]$ git checkout master
```

- Elimine su bifurcación local:

```
[student@workstation D0288-apps]$ git branch -d your-branch
```

- Elimine la bifurcación remota en su cuenta de GitHub personal:

```
[student@workstation D0288-apps]$ git push origin --delete your-branch
```

- Ahora puede reiniciar el ejercicio ejecutando su subcomando `start`:

```
[student@workstation D0288-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:

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 ~/D0288-apps  
[student@workstation D0288-apps]$ git commit -a -m 'Paused at step X.Y'  
[student@workstation D0288-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/v1/Git-Branching-What-a-Branch-Is>

Herramientas Git - Almacenamiento

<https://git-scm.com/book/en/v1/Git-Tools-Stashing>