





Join a community dedicated to learning open source

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



Network with tens of thousands of community members



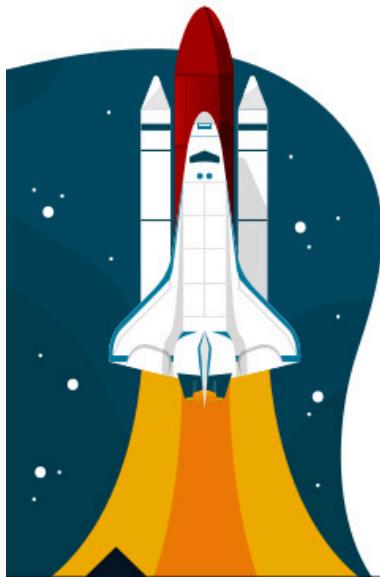
Engage in thousands of active conversations and posts



Join and interact with hundreds of certified training instructors



Unlock badges as you participate and accomplish new goals



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

Access free Red Hat training videos

Discover the latest Red Hat Training and Certification news

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

Join peers as you explore Red Hat products

Join the conversation learn.redhat.com



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

Red Hat OpenShift I: Containers & Kubernetes



OCP 4.10 DO180

Red Hat OpenShift I: Containers & Kubernetes

Edição 120220323

data de publicação 20220323

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

Copyright © 2021 Red Hat, Inc.

The contents of this course and all its modules and related materials, including handouts to audience members, are
Copyright © 2021 Red Hat, Inc.

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

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

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

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

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

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

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

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

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

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

All other trademarks are the property of their respective owners.

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

Convenções do documento	ix
Introdução	xi
DO180: Red Hat OpenShift I: Containers & Kubernetes	xi
Orientação para o ambiente da sala de aula	xii
Internacionalização	xvi
1. Introdução à tecnologia de contêiner	1
Visão geral da tecnologia de contêineres	2
Teste: Visão geral da tecnologia de contêineres	5
Visão geral da arquitetura do contêiner	9
Teste: Visão geral da arquitetura do contêiner	12
Visão geral do Kubernetes e OpenShift	14
Teste: Descrição do Kubernetes e do OpenShift	17
Exercício Guiado: Configuração do ambiente de sala de aula	19
Sumário	29
2. Criação de serviços conteinerizados	31
Provisionamento de serviços conteinerizados	32
Exercício Guiado: Criação de uma instância de banco de dados do MySQL	38
Uso de contêineres sem raiz	41
Exercício Guiado: Exploração de contêineres raiz e sem raiz	44
Laboratório Aberto: Criação de serviços conteinerizados	46
Sumário	50
3. Gerenciamento de contêineres	51
Gerenciamento do ciclo de vida de contêineres	52
Exercício Guiado: Gerenciamento de um contêiner MySQL	60
Anexação de armazenamento persistente aos contêineres	64
Exercício Guiado: Criar contêiner MySQL com banco de dados persistente	68
Acesso aos contêineres	71
Exercício Guiado: Carregamento do banco de dados	73
Laboratório Aberto: Gerenciamento de contêineres	76
Sumário	84
4. Gerenciamento de imagens de contêiner	85
Acesso aos registros	86
Teste: Trabalho com registros	92
Manipulação de imagens de contêiner	96
Exercício Guiado: Criação de uma imagem personalizada de contêiner do Apache	102
Laboratório Aberto: Gerenciamento de imagens	107
Sumário	115
5. Criação de imagens personalizadas de contêiner	117
Design para imagens personalizadas de contêiner	118
Teste: Abordagens para projetar imagens de contêiner	122
Criação de imagens personalizadas de contêiner com Containerfiles	124
Exercício Guiado: Criação de uma imagem básica de contêiner do Apache	130
Laboratório Aberto: Criação de imagens personalizadas de contêiner	134
Sumário	141
6. Implantação de aplicativos conteinerizados no OpenShift	143
Descrição da arquitetura do Kubernetes e do OpenShift	144
Teste: Descrição do Kubernetes e do OpenShift	151
Criação de recursos do Kubernetes	155
Exercício Guiado: Implantação de um servidor de banco de dados no OpenShift	168
Criação de rotas	173
Exercício Guiado: Exposição de um serviço como uma rota	177

Criação de aplicativos com Source-to-Image	182
Exercício Guiado: Criação de um aplicativo conteinerizado com Source-to-Image	192
Criação de aplicativos com o console da web do OpenShift	198
Exercício Guiado: Criação de um aplicativo com o console da web	204
Laboratório Aberto: Implantação de aplicativos conteinerizados no OpenShift	220
Sumário	224
7. Implantação de aplicativos com vários contêineres	225
Considerações sobre aplicativos de vários contêineres	226
Exercício Guiado: Implantação de aplicativo web e MySQL em contêineres Linux	231
Implantação de um aplicativo de vários contêineres no OpenShift	236
Exercício Guiado: Criação de um aplicativo no OpenShift.	238
Implantação de um aplicativo de vários contêineres no OpenShift usando um template ...	242
Exercício Guiado: Criação de um aplicativo com um template	249
Laboratório Aberto: Implantação de aplicativos com vários contêineres	253
Sumário	260
8. Solução de problemas de aplicativos conteinerizados	261
Solução de problemas de builds e implantações S2I	262
Exercício Guiado: Solução de problemas de uma build do OpenShift	268
Solução de problemas de aplicativos conteinerizados	276
Exercício Guiado: Configuração de logs de contêiner do Apache para depuração	283
Laboratório Aberto: Solução de problemas de aplicativos conteinerizados	286
Sumário	297
9. Revisão abrangente	299
Revisão abrangente	300
Laboratório Aberto: Conteinerização e implantação de um aplicativo de software	302
A. Implementação de arquitetura de microserviços	313
Implementação de arquiteturas de microserviços	314
Exercício Guiado: Refatoração do aplicativo To Do List	319
Sumário	324
B. Criação de uma conta no GitHub	325
Criação de uma conta no GitHub	326
C. Criação de uma conta no Quay	329
Criação de uma conta no Quay	330
Visibilidade de repositórios	332
D. Criação de uma conta da Red Hat	335
Criação de uma conta da Red Hat	336
E. Comandos git úteis	339
Comandos git	340

Convenções do documento



Referências

As "Referências" descrevem onde encontrar documentação externa relevante para um assunto.



nota

As "Observações" são dicas, atalhos ou abordagens alternativas para a tarefa em questão. Ignorar uma observação não deve acarretar em consequências negativas, mas você pode acabar perdendo algum truque que facilitará a sua vida.



Importante

As caixas "Importante" apresentam detalhes de itens que são facilmente negligenciados: alterações de configuração que se aplicam apenas à sessão atual ou serviços que devem ser reiniciados antes da aplicação de uma atualização. Ignorar uma caixa identificada como "Importante" não causa perda de dados, mas pode gerar irritação e frustração.



Cuidado

Os "Avisos" não devem ser ignorados. Ignorar avisos provavelmente causará a perda de dados.

Introdução

DO180: Red Hat OpenShift I: Containers & Kubernetes

DO180: Red Hat OpenShift I: Containers & Kubernetes é um curso prático para criar, implantar e gerenciar contêineres usando Podman, Kubernetes e o Red Hat OpenShift Container Platform.

Um dos locatários-chave do movimento DevOps é integração e implantação contínuas. Os contêineres tornaram-se uma tecnologia-chave para a configuração e implantação de aplicativos e microserviços. O Red Hat OpenShift Container Platform é uma implementação do Kubernetes, um sistema de orquestração de contêineres.

Objetivos do curso

- Demonstrar conhecimento sobre o ecossistema de contêineres.
- Gerenciar contêineres Linux usando Podman.
- Implantar contêineres em um cluster do Kubernetes usando o OpenShift Container Platform.
- Demonstrar design básico de contêineres e a habilidade de criar imagens de contêiner.
- Implantar uma arquitetura baseada em contêineres usando o conhecimento sobre contêineres, Kubernetes, e OpenShift.

Público-alvo

- Administradores de sistema
- Desenvolvedores
- Líderes de TI e arquitetos de infraestrutura

Pré-requisitos

Os alunos devem atender a um ou mais dos seguintes pré-requisitos:

- Conseguir usar uma sessão de terminal Linux e emitir comandos de sistema operacional. Uma certificação RHCSA é recomendada, mas não exigida.
- Ter experiência com arquiteturas de aplicativos web e suas respectivas tecnologias.

Orientação para o ambiente da sala de aula

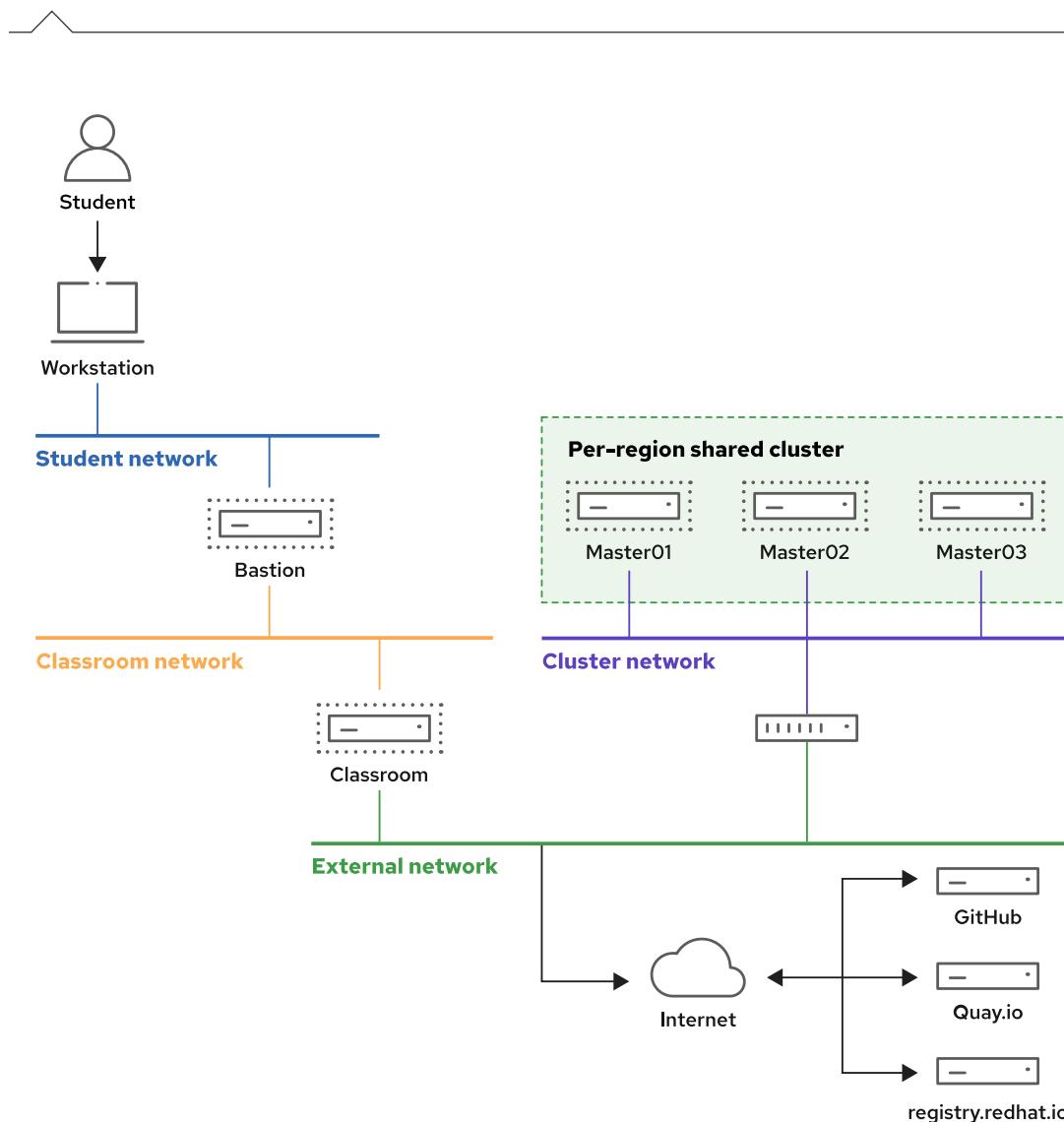


Figura 0.1: Ambiente de sala de aula

Neste curso, o principal sistema de computadores usado para atividades práticas de aprendizagem é a **workstation**. Esta é uma máquina virtual (VM) chamada `workstation.lab.example.com`.

Todas as máquinas possuem uma conta de usuário padrão, **student**, com a senha **student**. A senha do **root** em todos os sistemas de alunos é **redhat**.

Máquinas da sala de aula

Nome da máquina	Endereços IP	Função
workstation.lab.example.com	172.25.250.9	Workstation gráfica usada para administração do sistema.
classroom.example.com	172.25.254.254	Roteador que vincula a rede Classroom à internet.
bastion.lab.example.com	172.25.252.1	Roteador que vincula a rede Student à rede Classroom.

Vários sistemas na sala de aula oferecem serviços de suporte. Dois servidores, `content.example.com` e `materials.example.com`, são fontes de materiais de software e laboratório usados em atividades práticas. Informações sobre como usar esses servidores são fornecidas nas instruções dessas atividades.

Os alunos usam a máquina `workstation` para acessar um cluster do OpenShift compartilhado hospedado externamente. Os alunos não têm privilégios de administrador de cluster no cluster, mas isso não é necessário para concluir o conteúdo DO180.

Os alunos recebem uma conta em um cluster do OpenShift 4 compartilhado quando eles provisionam seus ambientes na interface Red Hat Online Learning. As informações do cluster, como o endpoint da API e a ID do cluster, bem como o nome de usuário e a senha, são apresentadas aos alunos quando provisionam seus ambientes.

Os alunos também têm acesso a um servidor MySQL e Nexus hospedados pelo cluster do OpenShift ou por um provedor externo. As atividades práticas neste curso fornecem instruções para acessar esses servidores quando necessário.

As atividades práticas no DO180 também exigem que os alunos tenham contas pessoais em dois serviços públicos e com Internet gratuitos: GitHub e Quay.io. Se ainda não tiverem essas contas (consulte o Apêndice), os alunos precisam criá-las e verificar seus acessos fazendo login nesses serviços antes de começar a aula.

Computadores remotos são atribuídos aos alunos em uma sala do Red Hat Online Learning. Eles são acessados em um aplicativo web hospedado em `rol.redhat.com` [<http://rol.redhat.com>]. Os alunos devem fazer login nesse site usando suas credenciais de usuário do Red Hat Customer Portal.

Controle de máquinas virtuais

As máquinas virtuais no ambiente de sala de aula são controladas por meio de uma página da web. O estado de cada máquina virtual na sala de aula é exibido na página na guia Lab Environment.

Estados da máquina

Estado da máquina virtual	Descrição
active	A máquina virtual está em execução e disponível (ou logo estará, ao inicializar).
interrompida	A máquina virtual está completamente desligada.

Estado da máquina virtual	Descrição
construção	A criação inicial da máquina virtual está sendo realizada.

Dependendo do estado da máquina, algumas das ações a seguir estarão disponíveis para você.

Ações de sala de aula/máquina

Botão ou ação	Descrição
CREATE	Criar a sala de aula do ROL. Cria todas as máquinas virtuais necessárias para a sala de aula e as inicia. Isso levará vários minutos para ser concluído.
DELETE	Excluir a sala de aula ROL. Destruirá todas as máquinas virtuais na sala de aula. Cuidado: qualquer trabalho gerado nesses discos será perdido.
START	Iniciar todas as máquinas virtuais na sala de aula.
OPEN CONSOLE	Abrir uma nova guia no navegador e conectar-se ao console da máquina virtual. Os alunos podem fazer login diretamente na máquina virtual e executar comandos. Na maioria dos casos, os alunos devem fazer login na máquina virtual workstation e usar ssh para se conectar a outras máquinas virtuais.
Iniciar	Iniciar (ligar) a máquina virtual.
Desligar	Desligar cuidadosamente a máquina virtual, preservando o conteúdo de seu disco.
Desligar	Desligar de maneira forçada a máquina virtual, preservando o conteúdo de seu disco. Isso equivale a remover a energia de uma máquina física.
Redefinir	Desligar de maneira forçada a máquina virtual e redefinir o disco para seu estado inicial. Cuidado: os trabalhos gerados nos discos serão perdidos.

Se você desejar retornar o ambiente de sala de aula para o estado original no início do curso, clique em **DELETE** para remover todo o ambiente de sala de aula. Depois que o laboratório for excluído, clique em **CREATE** para provisionar um novo conjunto de sistemas de sala de aula.



Cuidado

Não é possível desfazer a operação **DELETE**. Qualquer trabalho realizado no ambiente de sala de aula até esse ponto será perdido.

Temporizador de interrupção automática

O ingresso no ambiente de aprendizado on-line da Red Hat garante aos alunos tempo para o uso do computador. Para ajudar a economizar tempo alocado do computador, a sala de aula ROL tem

um cronômetro de contagem regressiva associado que desliga o ambiente de sala de aula quando o tempo acaba.

Para ajustar o temporizador, clique em [+] para adicionar uma hora ao temporizador. Observe que há um tempo máximo de doze horas.

Internacionalização

Seleção de idioma de acordo com o usuário

Os usuários talvez prefiram usar um idioma no ambiente da área de trabalho diferente do idioma padrão do sistema. Eles também podem usar um layout de teclado ou método de entrada diferente em suas contas.

Configurações de idioma

No ambiente GNOME de área de trabalho, o usuário poderá ser solicitado a definir o idioma e o método de entrada de preferência ao fazer o primeiro login. Caso isso não aconteça, a maneira mais fácil de um usuário ajustar as configurações preferenciais de idioma e método de entrada é usar o aplicativo **Region & Language**.

É possível iniciar o aplicativo de duas maneiras. Você pode executar o comando `gnome-control-center region` em uma janela de terminal, ou na barra superior, no menu do sistema no canto direito e selecionar o botão de configurações (que tem um ícone de chave inglesa com chave de fenda) na parte inferior esquerda do menu.

Na janela que é exibida, selecione **Region & Language**. Clique na caixa **Language** e selecione o idioma desejado na lista exibida. Isso também atualiza a configuração de **Formats** para o padrão do idioma escolhido. Na próxima vez que você fizer login, essas alterações entrarão em vigor.

Essas configurações afetam o ambiente GNOME de área de trabalho e todos os aplicativos iniciados dentro dele, como o `gnome-terminal`. No entanto, por padrão, elas não se aplicarão à conta caso sejam acessadas por meio de um login `ssh` a partir de um sistema remoto ou um login baseado em texto em um console virtual (como `tty5`).



nota

Você pode fazer com que o ambiente de shell use a mesma configuração de `LANG` que seu ambiente gráfico, mesmo quando fizer login por meio de um console virtual baseado em texto ou por `ssh`. Uma forma de fazer isso é colocar um código semelhante ao código a seguir no seu arquivo `~/.bashrc`. Este código de exemplo definirá o idioma usado no texto de login para corresponder àquele atualmente definido para o ambiente GNOME de área de trabalho do usuário:

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

Chinês, coreano, japonês e outros idiomas com conjuntos de caracteres não latinos podem não ser exibidos corretamente nos consoles virtuais baseados em texto.

Comandos individuais podem ser feitos para usar outro idioma definindo a variável `LANG` na linha de comando:

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

Os comandos subsequentes voltarão a usar o idioma padrão do sistema para saída. O comando `locale` pode ser usado para determinar o valor atual de `LANG` e de outras variáveis de ambiente relacionadas.

Configurações de método de entrada

O GNOME 3 no Red Hat Enterprise Linux 7 ou posterior usa automaticamente o sistema de seleção de método de entrada `IBus`, o que facilita a alteração rápida dos layouts de teclado e dos métodos de entrada.

O aplicativo `Region & Language` também pode ser usado para ativar métodos de entrada alternativos. Na janela do aplicativo `Region & Language`, a caixa [`Input Sources`] exibe os métodos de entrada disponíveis no momento. Por padrão, [English (US)] pode ser o único método disponível. Realce [English (US)] e clique no ícone de [keyboard] para ver o layout de teclado atual.

Para adicionar outro método de entrada, clique no botão [+] na parte inferior esquerda da janela [`Input Sources`]. A janela [`Add an Input Source`] será exibida. Selecione o idioma e, em seguida, o método de entrada ou o layout do teclado de sua preferência.

Quando mais de um método de entrada for configurado, o usuário poderá alternar rapidamente entre eles digitando [`Super+Space`] (às vezes, chamado de [`Windows+Space`]). Um indicador de status também será exibido na barra superior do GNOME, que tem duas funções: indica qual método de entrada está ativo e atua como um menu que pode ser usado para alternar entre métodos de entrada ou selecionar recursos avançados de métodos de entrada mais complexos.

Alguns dos métodos são marcados com engrenagens indicando que eles possuem opções de configurações e recursos avançados. Por exemplo, o método de entrada japonês, [Japanese (Kana Kanji)], permite ao usuário editar previamente um texto em latim e usar as teclas de [seta para baixo] e [seta para cima] a fim de selecionar os caracteres corretos a serem usados.

Falantes do inglês também podem achar essa técnica útil. Por exemplo, em [English (United States)], há o layout de teclado [English (international AltGr dead keys)], que trata [`AltGr`] (ou o [`Alt`] da direita) em um teclado de PC de 104/105 como uma tecla "shift secundária" modificadora e tecla de ativação inativa para digitação de caracteres adicionais. Também há o Dvorak e outros layouts alternativos disponíveis.



nota

Qualquer caractere Unicode poderá ser inserido no ambiente GNOME de área de trabalho se você conhecer o ponto de código Unicode do caractere. Digite [`Ctrl +Shift+U`], seguido do ponto de código. Depois de digitar [`Ctrl+Shift+U`], um u sublinhado será exibido para indicar que o sistema está aguardando a entrada do ponto de código Unicode.

Por exemplo, a letra grega lambda minúscula tem o ponto de código U+03BB e pode ser inserida digitando [`Ctrl+Shift+U`], depois `03BB` e, por fim, [`Enter`].

Configurações de idioma padrão em todo o sistema

O idioma padrão do sistema está definido como inglês dos EUA, que usa a codificação Unicode UTF-8 como seu conjunto de caracteres (`en_US.utf8`), mas isso pode ser alterado durante ou após a instalação.

Na linha de comando, o usuário `root` pode alterar as configurações de local do sistema com o comando `localectl`. Se `localectl` for executado sem erros, ele exibirá as atuais configurações de local em todo o sistema.

Para definir o idioma padrão para todo o sistema, execute o comando `localectl set-locale LANG=locale`, em que `locale` é o valor apropriado da variável de ambiente `LANG` na tabela "Referência de códigos de idioma" deste capítulo. A alteração será colocada em vigor para os usuários no próximo login e será armazenada em `/etc/locale.conf`.

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

No GNOME, um usuário administrativo pode alterar essa configuração em `Region & Language` clicando no botão [Login Screen] no canto superior direito da janela. Alterar [Language] na tela gráfica de login também ajustará a configuração do idioma padrão de todo o sistema armazenada no arquivo de configuração `/etc/locale.conf`.



Importante

Os consoles virtuais baseados em texto, como `tty4`, têm recursos mais limitados para exibir fontes do que os terminais de um console virtual que executa um ambiente gráfico, ou do que os pseudoterminais, para sessões `ssh`. Por exemplo, os caracteres do chinês, do coreano e do japonês podem não ser exibidos como esperado em um console virtual baseado em texto. Por isso, você deve considerar usar o inglês ou outro idioma com um conjunto de caracteres latinos como padrão para o sistema.

Da mesma forma, consoles virtuais baseados em texto são mais limitados em relação aos métodos de entrada compatíveis, e isso é gerenciado separadamente do ambiente gráfico de área de trabalho. As configurações de entrada globais disponíveis podem ser definidas por meio do `localectl` tanto para consoles virtuais baseados em texto quanto para o ambiente gráfico. Consulte as páginas do man `localectl(1)` e `vconsole.conf(5)` para obter mais informações.

Pacotes de idiomas

Pacotes RPM especiais chamados de langpacks instalaram pacotes de idiomas que adicionam suporte para idiomas específicos. Esses langpacks usam dependências para instalar automaticamente pacotes RPM adicionais que contêm localizações, dicionários e traduções para outros pacotes de software do seu sistema.

Para listar os langpacks que estão instalados e que podem ser instalados, use `yum list langpacks-*`:

```
[root@host ~]# yum list langpacks-*
Updating Subscription Management repositories.
Updating Subscription Management repositories.
Installed Packages
langpacks-en.noarch      1.0-12.el8      @AppStream
```

Introdução

```
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 adicionar suporte a idiomas, instale os langpacks apropriados. Por exemplo, o seguinte comando adiciona suporte para francês:

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

Use `yum repoquery --whatplements langpacks-fr` para determinar quais pacotes RPM podem ser instalados por um pacote de idiomas:

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

**Importante**

Os langpacks usam dependências fracas de RPM para instalar pacotes complementares somente quando o pacote principal que precisa delas também for instalado.

Por exemplo, ao instalar `langpacks-fr` como mostrado nos exemplos anteriores, o pacote `mythes-fr` só será instalado se o dicionário de sinônimos `mythes` também for instalado no sistema.

E se `mythes` for instalado posteriormente nesse sistema, o pacote `mythes-fr` também será instalado automaticamente devido à fraca dependência do pacote `langpacks-fr` já instalado.

**Referências**

Páginas do man `locale(7)`, `localectl(1)`, `locale.conf(5)`, `vconsole.conf(5)`, `unicode(7)` e `utf-8(7)`

As conversões entre os nomes dos layouts do ambiente gráfico de trabalho X11 e seus nomes em `localectl` podem ser encontradas no arquivo `/usr/share/X11/xkb/rules/base.lst`.

Referência de códigos de idioma


nota

Esta tabela pode não refletir todos os langpacks disponíveis no sistema. Use `yum info langpacks-SUFFIX` para obter mais informações sobre um langpack específico.

Códigos de idioma

Idioma	Sufixo dos langpacks	Valor \$LANG
Inglês (EUA)	en	en_US.utf8
Assamês	as	as_IN.utf8
Bengalês	bn	bn_IN.utf8
Chinês (simplificado)	zh_CN	zh_CN.utf8
Chinês (tradicional)	zh_TW	zh_TW.utf8
Francês	fr	fr_FR.utf8
Alemão	de	de_DE.utf8
Gujarati	gu	gu_IN.utf8
Híndi	hi	hi_IN.utf8
Italiano	it	it_IT.utf8
Japonês	ja	ja_JP.utf8
Canarês	kn	kn_IN.utf8
Coreano	ko	ko_KR.utf8
Malaiala	ml	ml_IN.utf8
Marata	mr	mr_IN.utf8
Oriya	ou	or_IN.utf8
Português (Brasil)	pt_BR	pt_BR.utf8
Punjabi	pa	pa_IN.utf8
Russo	ru	ru_RU.utf8
Espanhol	es	es_ES.utf8
Tâmil	ta	ta_IN.utf8
Telugo	te	te_IN.utf8

capítulo 1

Introdução à tecnologia de contêiner

Meta

Descrever como os aplicativos podem ser executados em contêineres orquestrados pelo Red Hat OpenShift Container Platform.

Objetivos

- Descrever a diferença entre os aplicativos de contêiner e implantações tradicionais.
- Descrever os conceitos básicos da arquitetura de contêineres.
- Descrever os benefícios da orquestração de aplicativos e da OpenShift Container Platform.

Seções

- Visão geral da tecnologia de contêineres (e teste)
- Visão geral da arquitetura de contêineres (e teste)
- Visão geral do Kubernetes e OpenShift (e teste)

Visão geral da tecnologia de contêineres

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de descrever a diferença entre os aplicativos de contêiner e implantações tradicionais.

Aplicativos conteinerizados

Os aplicativos de software normalmente dependem de outras bibliotecas, arquivos de configuração ou serviços fornecidos pelo ambiente de tempo de execução. O ambiente de tempo de execução tradicional para um aplicativo de software é um host físico ou uma máquina virtual, e as dependências do aplicativo são instaladas como parte do host.

Por exemplo, considere um aplicativo Python que exige acesso a uma biblioteca compartilhada comum que implementa o protocolo TLS. Tradicionalmente, um administrador de sistema instala o pacote necessário que fornece a biblioteca compartilhada antes de instalar o aplicativo Python.

A principal desvantagem de um aplicativo de software implantado tradicionalmente é que as dependências do aplicativo estão unidas ao ambiente de tempo de execução.

Um aplicativo pode sofrer danos quando uma atualização ou correção forem aplicadas ao sistema operacional (SO) de base.

Por exemplo, uma atualização do SO para a biblioteca compartilhada TLS remove a compatibilidade do protocolo TLS 1.0. Isso causa danos ao aplicativo Python implantado porque ele é gravado para usar o protocolo TLS 1.0 para solicitações de rede. Isso força o administrador do sistema a reverter a atualização do SO para manter o aplicativo em execução, impedindo que outros aplicativos usem os benefícios do pacote atualizado.

Assim, para uma empresa que esteja desenvolvendo aplicativos de software tradicionais, talvez seja necessário um conjunto completo de testes para garantir que nenhuma atualização do SO afete aplicativos em execução no host.

Além disso, um aplicativo implantado tradicionalmente deve ser interrompido antes de atualizar as dependências associadas. Para minimizar o tempo de inatividade do aplicativo, as organizações projetam e implementam sistemas complexos para fornecer alta disponibilidade de seus aplicativos. A manutenção de vários aplicativos em um único host muitas vezes se torna complexa, e existe a possibilidade de as implantações ou atualizações causarem danos a um dos aplicativos da organização.

A *Figura 1.1* descreve a diferença entre aplicativos em execução como contêineres e aplicativos em execução no sistema operacional do host.

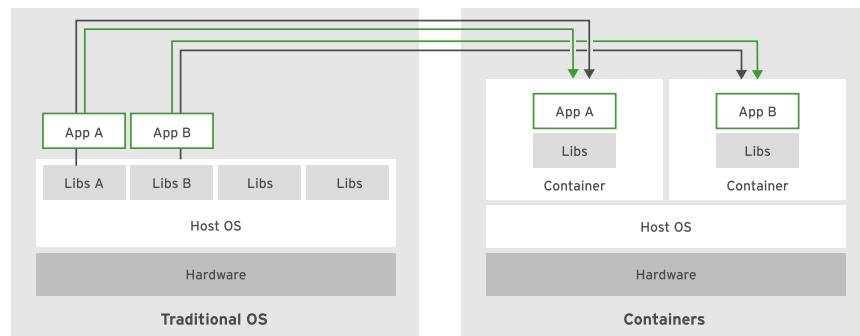


Figura 1.1: Diferenças entre o contêiner e o sistema operacional

Como alternativa, um aplicativo de software pode ser implantado usando um contêiner.

Um contêiner é um conjunto de um ou mais processos que são isolados do restante do sistema.

Os contêineres oferecem muitos dos mesmos benefícios das máquinas virtuais, como segurança, armazenamento e isolamento de rede. Os contêineres exigem muito menos recursos de hardware e têm inicialização e encerramento rápidos. Eles também isolam as bibliotecas e os recursos de tempo de execução (como a CPU e o armazenamento) de um aplicativo para minimizar o impacto de qualquer atualização no SO do host, conforme descrito na *Figura 1.1*.

O uso de contêineres não só aumenta a eficiência, a flexibilidade e a reusabilidade dos aplicativos de host, mas também a portabilidade do aplicativo. O Open Container Initiative (OCI) fornece um conjunto de padrões de mercado que define uma especificação de tempo de execução de contêiner e uma especificação de imagem de contêiner. A especificação da imagem define o formato do pacote de arquivos e metadados que formam uma imagem de contêiner. Quando você cria um aplicativo como uma imagem de contêiner que está em conformidade com o padrão OCI você pode usar qualquer mecanismo de contêiner que esteja em conformidade com OCI para executar o aplicativo.

Há muitos mecanismos de contêiner disponíveis para gerenciar e executar contêineres individuais, incluindo Rocket, Drawbridge, LXC, Docker e Podman. O Podman está disponível no Red Hat Enterprise Linux 7.6 e posterior, e é usado neste curso para iniciar, gerenciar e encerrar contêineres individuais.

Veja abaixo outras vantagens importantes para o uso de contêineres:

Baixa área de ocupação do hardware

Os contêineres usam recursos internos do SO para criar um ambiente isolado em que os recursos são gerenciados usando recursos do SO, como namespaces e cgroups. Essa abordagem minimiza a quantidade de sobrecarga de CPU e na memória em comparação a um hipervisor de máquina virtual. A execução de um aplicativo em uma VM é uma maneira de criar isolamento a partir do ambiente de execução, mas isso requer uma camada pesada de serviços para suportar o mesmo isolamento de baixa área de ocupação de hardware que é oferecido por contêineres.

Isolamento de ambiente

Os contêineres funcionam em um ambiente fechado onde as alterações feitas no SO de host ou outros aplicativos não afetam o contêiner. Como as bibliotecas necessárias para um contêiner são autocontidas, o aplicativo pode ser executado sem interrupções. Por exemplo, cada aplicativo pode existir em seu próprio contêiner com seu próprio conjunto de bibliotecas. Uma atualização feita em um contêiner não afeta outros contêineres.

Implantação rápida

Os contêineres são implantados rapidamente porque não há necessidade de instalar todo o sistema operacional subjacente. Normalmente, para dar suporte ao isolamento, uma nova instalação do SO é necessária em um host físico ou VM, e qualquer atualização simples pode precisar de uma reinicialização total do SO. A reinicialização de um contêiner não exige a interrupção de nenhum serviço no SO de host.

Implantação em múltiplos ambientes

Em um cenário tradicional de implantação usando um único host, qualquer diferença de ambiente pode interromper o aplicativo. Ao usar contêineres, no entanto, todas as dependências de aplicativo e configurações de ambiente são encapsuladas na imagem do contêiner.

Reusabilidade

O mesmo contêiner pode ser usado sem a necessidade de configurar um SO completo. Por exemplo, o mesmo contêiner de banco de dados que fornece um serviço de banco de dados de produção pode ser usado por cada desenvolvedor para criar um banco de dados de desenvolvimento durante o desenvolvimento do aplicativo. Ao usar contêineres, não há mais a necessidade de manter servidores de bancos de dados de produção e desenvolvimento separados. Uma única imagem de contêiner é usada para criar instâncias do serviço de banco de dados.

Frequentemente, um aplicativo de software com todos os seus serviços dependentes (banco de dados, mensagem, sistemas de arquivo) é feito para ser executado em um único contêiner. Isso pode levar aos mesmos problemas associados às implantações tradicionais de software para máquinas virtuais ou hosts físicos. Nessas instâncias, uma implantação de vários contêineres pode ser mais adequada.

Além disso, os contêineres são uma abordagem ideal ao usar microsserviços para o desenvolvimento de aplicativos. Cada serviço é encapsulado em um ambiente de contêiner leve e confiável que pode ser implantado em um ambiente de produção ou desenvolvimento. A coleta de serviços em contêineres exigidos por um aplicativo pode ser hospedada em uma única máquina, eliminando a necessidade de gerenciar uma máquina para cada serviço.

Em contrapartida, muitos aplicativos não são adequados para um ambiente conteinerizado. Por exemplo, alguns aplicativos que acessam informações de hardware de baixo nível, como memória, sistemas de arquivo e dispositivos, podem não ser confiáveis devido às limitações do contêiner.



Referências

Home - Open Containers Initiative

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

► Teste

Visão geral da tecnologia de contêineres

Escolha as respostas corretas para as seguintes perguntas:

- 1. **Quais duas opções são exemplos de aplicativos de software que podem ser executados em um contêiner? (Escolha duas opções.)**
- a. Um aplicativo Python baseado em banco de dados acessando serviços como um banco de dados do MySQL, um servidor de protocolo de transferência de arquivo (FTP) e um servidor web em um único host físico.
 - b. Um aplicativo Java Enterprise Edition com um banco de dados Oracle e um agente de mensagens em execução em uma única VM.
 - c. Uma ferramenta de monitoramento de E/S responsável por analisar o tráfego e a transferência de dados de bloqueio.
 - d. Uma ferramenta de aplicativo de despejo de memória capaz de fazer snapshots de todos os caches da CPU de memória para fins de depuração.
- 2. **Entre as opções a seguir, quais são os dois casos de uso mais adequados para contêineres? (Escolha duas opções.)**
- a. Um provedor de software precisa distribuir software que possa ser reutilizado por outras empresas de maneira rápida e sem erros.
 - b. Uma empresa está implantando aplicativos em um host físico e gostaria de melhorar seu desempenho usando contêineres.
 - c. Os desenvolvedores de uma empresa precisam de um ambiente descartável que imite o ambiente de produção para que possam testar rapidamente o código que estão desenvolvendo.
 - d. Uma empresa do setor financeiro está implementando uma ferramenta de análise de risco que faz uso intensivo da CPU em seus próprios contêineres para minimizar o número de processadores necessários.

- 3. Uma empresa está migrando seus aplicativos PHP e Python executados no mesmo host para uma nova arquitetura. Devido a políticas internas, ambas estão usando um conjunto de bibliotecas compartilhadas e personalizadas do SO, mas a atualização mais recente aplicada a elas, como consequência de uma solicitação da equipe de desenvolvimento em Python, danificou o aplicativo PHP. Quais as duas arquiteturas que ofereceriam o melhor suporte para ambos os aplicativos? (Escolha duas opções.)
- a. Implante cada aplicativo em diferentes VMs e aplique as bibliotecas compartilhadas e personalizadas individualmente em cada host de VM.
 - b. Implante cada aplicativo em diferentes contêineres e aplique as bibliotecas compartilhadas e personalizadas individualmente em cada host de contêiner.
 - c. Implante cada aplicativo em diferentes VMs e aplique as bibliotecas compartilhadas e personalizadas em todos os hosts de VM.
 - d. Implante cada aplicativo em diferentes contêineres e aplique as bibliotecas compartilhadas e personalizadas em todos os contêineres.
- 4. Quais três tipos de aplicativos podem ser empacotados como contêineres para consumo imediato? (Escolha três opções.)
- a. Um hipervisor de máquina virtual
 - b. Um software de blog, como o WordPress
 - c. Um banco de dados
 - d. Uma ferramenta local de recuperação de sistema de arquivos
 - e. Um servidor web

► Solução

Visão geral da tecnologia de contêineres

Escolha as respostas corretas para as seguintes perguntas:

- 1. **Quais duas opções são exemplos de aplicativos de software que podem ser executados em um contêiner? (Escolha duas opções.)**
- a. Um aplicativo Python baseado em banco de dados acessando serviços como um banco de dados do MySQL, um servidor de protocolo de transferência de arquivo (FTP) e um servidor web em um único host físico.
 - b. Um aplicativo Java Enterprise Edition com um banco de dados Oracle e um agente de mensagens em execução em uma única VM.
 - c. Uma ferramenta de monitoramento de E/S responsável por analisar o tráfego e a transferência de dados de bloqueio.
 - d. Uma ferramenta de aplicativo de despejo de memória capaz de fazer snapshots de todos os caches da CPU de memória para fins de depuração.
- 2. **Entre as opções a seguir, quais são os dois casos de uso mais adequados para contêineres? (Escolha duas opções.)**
- a. Um provedor de software precisa distribuir software que possa ser reutilizado por outras empresas de maneira rápida e sem erros.
 - b. Uma empresa está implantando aplicativos em um host físico e gostaria de melhorar seu desempenho usando contêineres.
 - c. Os desenvolvedores de uma empresa precisam de um ambiente descartável que imite o ambiente de produção para que possam testar rapidamente o código que estão desenvolvendo.
 - d. Uma empresa do setor financeiro está implementando uma ferramenta de análise de risco que faz uso intensivo da CPU em seus próprios contêineres para minimizar o número de processadores necessários.

- 3. Uma empresa está migrando seus aplicativos PHP e Python executados no mesmo host para uma nova arquitetura. Devido a políticas internas, ambas estão usando um conjunto de bibliotecas compartilhadas e personalizadas do SO, mas a atualização mais recente aplicada a elas, como consequência de uma solicitação da equipe de desenvolvimento em Python, danificou o aplicativo PHP. Quais as duas arquiteturas que ofereceriam o melhor suporte para ambos os aplicativos? (Escolha duas opções.)
- a. Implante cada aplicativo em diferentes VMs e aplique as bibliotecas compartilhadas e personalizadas individualmente em cada host de VM.
 - b. Implante cada aplicativo em diferentes contêineres e aplique as bibliotecas compartilhadas e personalizadas individualmente em cada host de contêiner.
 - c. Implante cada aplicativo em diferentes VMs e aplique as bibliotecas compartilhadas e personalizadas em todos os hosts de VM.
 - d. Implante cada aplicativo em diferentes contêineres e aplique as bibliotecas compartilhadas e personalizadas em todos os contêineres.
- 4. Quais três tipos de aplicativos podem ser empacotados como contêineres para consumo imediato? (Escolha três opções.)
- a. Um hipervisor de máquina virtual
 - b. Um software de blog, como o WordPress
 - c. Um banco de dados
 - d. Uma ferramenta local de recuperação de sistema de arquivos
 - e. Um servidor web

Visão geral da arquitetura do contêiner

Objetivos

Depois de concluir esta seção, você deverá ser capaz de:

- Descrever a arquitetura de contêineres do Linux.
- Descrever a ferramenta podman para o gerenciamento de contêineres.

Introdução ao histórico do contêiner

Os contêineres ganharam popularidade rapidamente nos últimos anos. No entanto, a tecnologia por trás dos contêineres existe há, relativamente, bastante tempo. Em 2001, o Linux apresentou um projeto chamado VServer. O VServer foi a primeira tentativa de executar conjuntos completos de processos dentro de um único servidor com alto grau de isolamento.

A partir do VServer, a ideia de processos isolados evoluiu e se tornou formalizada em relação aos seguintes recursos do kernel do Linux:

Namespaces

Um namespace isola recursos específicos do sistema que normalmente ficam visíveis para todos os processos. Dentro de um namespace, somente processos que forem membros desse namespace podem ver tais recursos. Os namespaces podem incluir recursos como interfaces de rede, a lista de ID de processo, pontos de montagem, recursos IPC e as informações de nome do host do sistema.

Grupos de controle (cgroups)

Os grupos de controle particionam conjuntos de processos e seus filhos em grupos para gerenciar e limitar os recursos consumidos por eles. Os grupos de controle estabelecem restrições na quantidade de recursos de sistema que os processos podem usar. Essas restrições impedem que um processo use muitos recursos no host.

Seccomp

Desenvolvido em 2005 e introduzido nos contêineres por volta de 2014, o Seccomp limita a maneira com que os processos podem usar as chamadas do sistema. O Seccomp define um perfil de segurança para processos, colocando na lista de permissões as chamadas do sistema, os parâmetros e os descritores de arquivos que eles podem usar.

SELinux

O SELinux (Security-Enhanced Linux) é um sistema de controle de acesso obrigatório para processos. O kernel do Linux usa o SELinux para proteger processos uns dos outros e para proteger o sistema de host de seus processos em execução. Os processos são executados como um tipo SELinux confinado e têm acesso limitado aos recursos do sistema de host.

Todas essas inovações e recursos se concentram ao redor de um conceito básico: permitir que os processos sejam executados isolados enquanto ainda acessam os recursos do sistema. Esse conceito é o fundamento da tecnologia de contêineres e a base para todas as implementações de contêineres. Atualmente, os contêineres são processos no kernel do Linux que fazem uso desses recursos de segurança a fim de criar um ambiente isolado. Esse ambiente proíbe que processos isolados utilizem incorretamente o sistema ou outros recursos do contêiner.

capítulo 1 | Introdução à tecnologia de contêiner

Um caso de uso comum de contêineres é ter várias réplicas para o mesmo serviço (como um servidor de banco de dados) no mesmo host. Cada réplica tem recursos isolados (sistema de arquivos, portas, memória), portanto, não há necessidade de o serviço lidar com o compartilhamento de recursos. O isolamento garante que um serviço defeituoso ou nocivo não afete outros serviços ou contêineres no mesmo host, nem no sistema subjacente.

Descrição da arquitetura de contêineres do Linux

Da perspectiva do kernel do Linux, um contêiner é um processo com restrições. No entanto, em vez de executar um único arquivo binário, um contêiner executa uma imagem. A imagem é um pacote de sistema de arquivos que contém todas as dependências necessárias para executar um processo: arquivos no sistema de arquivos, pacotes instalados, recursos disponíveis, processos em execução e módulos do kernel.

Como os arquivos executáveis são a base para a execução de processos, as imagens são a base para a execução de contêineres. Os contêineres em execução usam uma visualização imutável da imagem, permitindo que vários contêineres reutilizem a mesma imagem simultaneamente. Como as imagens são arquivos, elas podem ser gerenciadas por sistemas de controle de versão, aprimorando a automação no contêiner e o provisionamento de imagens.

As imagens do contêiner precisam estar disponíveis localmente para que o tempo de execução do contêiner as execute, elas geralmente são armazenadas e mantidas em um repositório de imagens. Um repositório de imagens é apenas um serviço (público ou privado) onde as imagens podem ser armazenadas, pesquisadas e recuperadas. Outros recursos fornecidos pelos repositórios de imagem são acesso remoto, metadados de imagem, autorização ou controle de versão da imagem.

Há muitos repositórios de imagens diferentes disponíveis, cada um oferecendo recursos diferentes:

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



nota

Este curso usa o registro de imagem pública Quay, para que os alunos possam operar com imagens sem se preocupar com interferir uns com os outros.

Gerenciamento de contêineres com Podman

Contêineres, imagens e registros de imagens precisam interagir uns com os outros. Por exemplo, você precisa ser capaz de criar imagens e colocá-las em registros de imagens. Você também precisa ser capaz de recuperar uma imagem do registro de imagens e criar um contêiner a partir dessa imagem.

O Podman é uma ferramenta open source para gerenciar contêineres e imagens de contêiner e interagir com registros de imagens. Ele oferece os seguintes recursos principais:

- Ele usa o formato de imagem especificado pelo Open Container Initiative [<https://www.opencontainers.org>] (OCI). Essas especificações definem um formato de imagem padrão, orientado pela comunidade e não proprietário.
- O Podman armazena imagens locais em um sistema de arquivos local. Isso evita arquitetura de servidor/cliente desnecessária ou ter daemons em execução na máquina local.
- O Podman segue os mesmos padrões de comando da CLI do Docker; portanto, não é necessário aprender a usar um novo conjunto de ferramentas.
- O Podman é compatível com o Kubernetes. O Kubernetes pode usar o Podman para gerenciar seus contêineres.



Referências

Red Hat Quay Container Registry

<https://quay.io>

Site do Podman

<https://podman.io/>

Open Container Initiative

<https://www.opencontainers.org>

► Teste

Visão geral da arquitetura do contêiner

Escolha as respostas corretas para as seguintes perguntas:

- ▶ 1. Entre as opções a seguir, quais são os três recursos do Linux usados para executar contêineres? (Escolha três opções.)
 - a. Namespaces
 - b. Gerenciamento de integridade
 - c. Security-Enhanced Linux
 - d. Grupos de controle

- ▶ 2. Qual das seguintes opções melhor descreve uma imagem de contêiner?
 - a. Uma imagem de máquina virtual na qual um contêiner será criado.
 - b. Uma referência de contêiner a partir do qual um contêiner será criado.
 - c. Um ambiente de tempo de execução onde um aplicativo será executado.
 - d. O arquivo de índice do contêiner usado por um registro.

- ▶ 3. Quais dos componentes a seguir são comuns em implementações de arquitetura de contêiner? (Escolha três opções.)
 - a. Tempo de execução do contêiner
 - b. Permissões do contêiner
 - c. Imagens do contêiner
 - d. Registros do contêiner

- ▶ 4. O que é um contêiner em relação ao kernel do Linux?
 - a. Uma máquina virtual.
 - b. Um processo isolado com acesso regulado a recursos.
 - c. Um conjunto de camadas do sistema de arquivos expostas pelo UnionFS.
 - d. Um serviço externo que fornece imagens do contêiner.

► Solução

Visão geral da arquitetura do contêiner

Escolha as respostas corretas para as seguintes perguntas:

- ▶ 1. **Entre as opções a seguir, quais são os três recursos do Linux usados para executar contêineres? (Escolha três opções.)**
 - a. Namespaces
 - b. Gerenciamento de integridade
 - c. Security-Enhanced Linux
 - d. Grupos de controle

- ▶ 2. **Qual das seguintes opções melhor descreve uma imagem de contêiner?**
 - a. Uma imagem de máquina virtual na qual um contêiner será criado.
 - b. Uma referência de contêiner a partir do qual um contêiner será criado.
 - c. Um ambiente de tempo de execução onde um aplicativo será executado.
 - d. O arquivo de índice do contêiner usado por um registro.

- ▶ 3. **Quais dos componentes a seguir são comuns em implementações de arquitetura de contêiner? (Escolha três opções.)**
 - a. Tempo de execução do contêiner
 - b. Permissões do contêiner
 - c. Imagens do contêiner
 - d. Registros do contêiner

- ▶ 4. **O que é um contêiner em relação ao kernel do Linux?**
 - a. Uma máquina virtual.
 - b. Um processo isolado com acesso regulado a recursos.
 - c. Um conjunto de camadas do sistema de arquivos expostas pelo UnionFS.
 - d. Um serviço externo que fornece imagens do contêiner.

Visão geral do Kubernetes e OpenShift

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Identificar as limitações dos contêineres do Linux e a necessidade de orquestração de contêineres.
- Descrever a ferramenta de orquestração de contêineres do Kubernetes.
- Descrever a Red Hat OpenShift Container Platform (RHOC).

Limitações dos contêineres

Os contêineres oferecem uma maneira fácil de empacotar e executar serviços. À medida que o número de contêineres gerenciados por uma organização aumenta, o trabalho de iniciá-los manualmente aumenta exponencialmente, juntamente com a necessidade de responder rapidamente às demandas externas.

Ao usar contêineres em um ambiente de produção, as empresas geralmente exigem:

- Comunicação fácil entre um grande número de serviços.
- Limites de recursos em aplicativos, independentemente do número de contêineres que os executam.
- Resposta a picos de uso de aplicativos para aumentar ou diminuir a execução de contêineres.
- Reação à deterioração do serviço.
- Lançamento gradual de uma nova versão para um conjunto de usuários.

Em geral, as empresas exigem uma tecnologia de orquestração de contêineres, porque os tempos de execução do contêiner (como o Podman) não atendem adequadamente aos requisitos acima.

Visão geral do Kubernetes

O Kubernetes é um serviço de orquestração que simplifica a implantação, o gerenciamento e o dimensionamento de aplicativos conteinerizados.

A menor unidade gerenciável no Kubernetes é um pod. Um pod é composto por um ou mais contêineres com seus recursos de armazenamento e endereço IP que representam um único aplicativo. O Kubernetes também usa pods para orquestrar os contêineres dentro dele e limitar seus recursos como uma única unidade.

Recursos do Kubernetes

O Kubernetes oferece os seguintes recursos na parte superior de uma infraestrutura de contêineres:

Descoberta de serviço e balanceamento de carga

O Kubernetes permite a comunicação entre serviços, atribuindo uma única entrada DNS a cada conjunto de contêineres. Dessa forma, o serviço solicitante só precisa saber o nome

DNS do destino, permitindo que o cluster altere a localização e o endereço IP do contêiner, deixando o serviço inalterado. Isso permite o balanceamento de carga da solicitação em todo o pool de contêineres que fornece o serviço. Por exemplo, o Kubernetes pode dividir de maneira uniforme as solicitações recebidas para um serviço MySQL, levando em conta a disponibilidade dos pods.

Escala horizontal

Os aplicativos podem aumentar ou diminuir a escala manual ou automaticamente com o conjunto de configurações com a interface de linha de comando do Kubernetes ou a interface do usuário da web.

Autorrecuperação

O Kubernetes pode usar verificações de integridade definidas pelo usuário para monitorar os contêineres a fim de reiniciar e reagendá-los em caso de falha.

Distribuição automatizada

O Kubernetes pode gradualmente distribuir atualizações para os contêineres do seu aplicativo enquanto verifica seu status. Se algo der errado durante a distribuição, o Kubernetes poderá reverter para a iteração anterior da implantação.

Gerenciamento de configurações e segredos

Você pode gerenciar as configurações e segredos dos seus aplicativos sem recriar contêineres. Os segredos de aplicativos podem ser nomes de usuários, senhas e pontos de extremidade de serviço; qualquer configuração que precise ser mantida em sigilo.

Operadores

Os operadores são aplicativos empacotados do Kubernetes que também trazem o conhecimento do ciclo de vida do aplicativo para o cluster do Kubernetes. Os aplicativos empacotados como operadores usam a API do Kubernetes para atualizar o estado do cluster, reagindo a mudanças no estado do aplicativo.

Visão geral do OpenShift

O Red Hat OpenShift Container Platform (RHOP) é um conjunto de serviços e componentes modulares desenvolvidos com base em uma infraestrutura de contêineres do Kubernetes. O RHOP adiciona as capacidades de fornecer uma plataforma PaaS de produção, como gerenciamento remoto, multilocação, monitoramento e auditoria, segurança aumentada, gerenciamento de ciclo de vida de aplicativo e interfaces de autosserviço para desenvolvedores.

A partir do Red Hat OpenShift v4, todos os hosts em um cluster do OpenShift usam o Red Hat Enterprise Linux CoreOS como sistema operacional subjacente.



nota

Ao longo deste curso, os termos RHOP e OpenShift são usados para se referir ao Red Hat OpenShift Container Platform.

Recursos do OpenShift

O OpenShift adiciona os seguintes recursos a um cluster do Kubernetes:

Fluxo de trabalho do desenvolvedor integrado

O RHOP integra um container registry integrado, pipelines CI/CD e S2I; uma ferramenta para criar artefatos a partir de repositórios de origem para imagens de contêiner.

Rotas

Exponha facilmente os serviços ao mundo exterior.

Métricas e registros

Inclui serviço de métricas integrado e de autoanálise e registros agregados.

Interface de usuário unificada

O OpenShift traz ferramentas unificadas e uma interface de usuário para gerenciar todos os recursos diferentes.



Referências

Orquestração de contêineres de nível de produção - Kubernetes

<https://kubernetes.io/>

OpenShift: Plataforma de aplicativo de contêineres da Red Hat, criada no

Docker e no Kubernetes

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

► Teste

Descrição do Kubernetes e do OpenShift

Escolha as respostas corretas para as seguintes perguntas:

- ▶ 1. **Entre as opções a seguir, quais são as três afirmações corretas em relação às limitações dos contêineres? (Escolha três opções.)**
 - a. Os contêineres são facilmente orquestrados em grande número.
 - b. A falta de automação aumenta o tempo de resposta aos problemas.
 - c. Os contêineres não gerenciam falhas de aplicativos dentro deles.
 - d. Os contêineres não têm balanceamento de carga.
 - e. Os contêineres são aplicativos empacotados isolados.

- ▶ 2. **Entre as opções a seguir, quais são as duas afirmações corretas em relação ao Kubernetes? (Escolha duas opções.)**
 - a. O Kubernetes é um contêiner.
 - b. O Kubernetes só pode usar contêineres do Docker.
 - c. O Kubernetes é um sistema de orquestração de contêineres.
 - d. O Kubernetes simplifica o gerenciamento, a implantação, e o dimensionamento de aplicativos conteinerizados.
 - e. É mais difícil manter aplicativos gerenciados em um cluster do Kubernetes.

- ▶ 3. **Entre as opções a seguir, quais são as três afirmações verdadeiras em relação ao Red Hat OpenShift v4? (Escolha três opções.)**
 - a. O OpenShift fornece recursos adicionais para uma infraestrutura do Kubernetes.
 - b. O Kubernetes e o OpenShift são mutuamente exclusivos.
 - c. Os hosts do OpenShift usam o Red Hat Enterprise Linux como sistema operacional de base.
 - d. O OpenShift simplifica o desenvolvimento incorporando uma tecnologia Source-to-Image e pipelines CI/CD.
 - e. O OpenShift simplifica o roteamento e o balanceamento de carga.

- ▶ 4. **Quais recursos o OpenShift oferece para ampliar os recursos do Kubernetes? (Escolha duas opções.)**
 - a. Operators e Operator Framework.
 - b. Encaminha para expor os serviços ao mundo exterior.
 - c. Um fluxo de trabalho do desenvolvedor integrado.
 - d. Autorrecuperação e verificações de integridade.

► Solução

Descrição do Kubernetes e do OpenShift

Escolha as respostas corretas para as seguintes perguntas:

- ▶ 1. Entre as opções a seguir, quais são as três afirmações corretas em relação às limitações dos contêineres? (Escolha três opções.)
 - a. Os contêineres são facilmente orquestrados em grande número.
 - b. A falta de automação aumenta o tempo de resposta aos problemas.
 - c. Os contêineres não gerenciam falhas de aplicativos dentro deles.
 - d. Os contêineres não têm balanceamento de carga.
 - e. Os contêineres são aplicativos empacotados isolados.

- ▶ 2. Entre as opções a seguir, quais são as duas afirmações corretas em relação ao Kubernetes? (Escolha duas opções.)
 - a. O Kubernetes é um contêiner.
 - b. O Kubernetes só pode usar contêineres do Docker.
 - c. O Kubernetes é um sistema de orquestração de contêineres.
 - d. O Kubernetes simplifica o gerenciamento, a implantação, e o dimensionamento de aplicativos conteinerizados.
 - e. É mais difícil manter aplicativos gerenciados em um cluster do Kubernetes.

- ▶ 3. Entre as opções a seguir, quais são as três afirmações verdadeiras em relação ao Red Hat OpenShift v4? (Escolha três opções.)
 - a. O OpenShift fornece recursos adicionais para uma infraestrutura do Kubernetes.
 - b. O Kubernetes e o OpenShift são mutuamente exclusivos.
 - c. Os hosts do OpenShift usam o Red Hat Enterprise Linux como sistema operacional de base.
 - d. O OpenShift simplifica o desenvolvimento incorporando uma tecnologia Source-to-Image e pipelines CI/CD.
 - e. O OpenShift simplifica o roteamento e o balanceamento de carga.

- ▶ 4. Quais recursos o OpenShift oferece para ampliar os recursos do Kubernetes? (Escolha duas opções.)
 - a. Operators e Operator Framework.
 - b. Encaminha para expor os serviços ao mundo exterior.
 - c. Um fluxo de trabalho do desenvolvedor integrado.
 - d. Autorrecuperação e verificações de integridade.

► Exercício Guiado

Configuração do ambiente de sala de aula

Neste exercício, você configurará a workstation para acessar toda a infraestrutura usada por este curso.

Resultados

Você deverá ser capaz de:

- Configurar a workstation para acessar um cluster do OpenShift, um registro de imagem de contêiner e um repositório Git usados em todo o curso.
- Ramificar o repositório dos aplicativos de exemplo deste curso para sua conta pessoal do GitHub.
- Clonar o repositório dos aplicativos de exemplo deste curso de sua conta pessoal do GitHub para sua máquina workstation .

Antes De Começar

Para realizar este exercício, certifique-se de que você tenha:

- Acesso ao curso DO180 no ambiente de aprendizado on-line do treinamento da Red Hat.
- Os parâmetros de conexão e uma conta de usuário do desenvolvedor para acessar um cluster do OpenShift gerenciado pela Red Hat Training.
- Uma conta pessoal e gratuita do GitHub. Se precisar se registrar no GitHub, consulte as instruções em *Apêndice B, Criação de uma conta no GitHub*.
- Uma conta pessoal e gratuita do Quay.io. Se precisar se registrar no Quay.io, consulte as instruções em *Apêndice C, Criação de uma conta no Quay*.
- Um token de acesso pessoal do GitHub.

Instruções

Antes de iniciar qualquer exercício, certifique-se de que você:

- 1. Prepare your GitHub access token.
- 1.1. Navegou até <https://github.com> usando um navegador e fez a autenticação.
 - 1.2. Na parte superior da página, clique no ícone do perfil, selecione o menu **Settings** e selecione **Developer settings** no painel esquerdo da página.

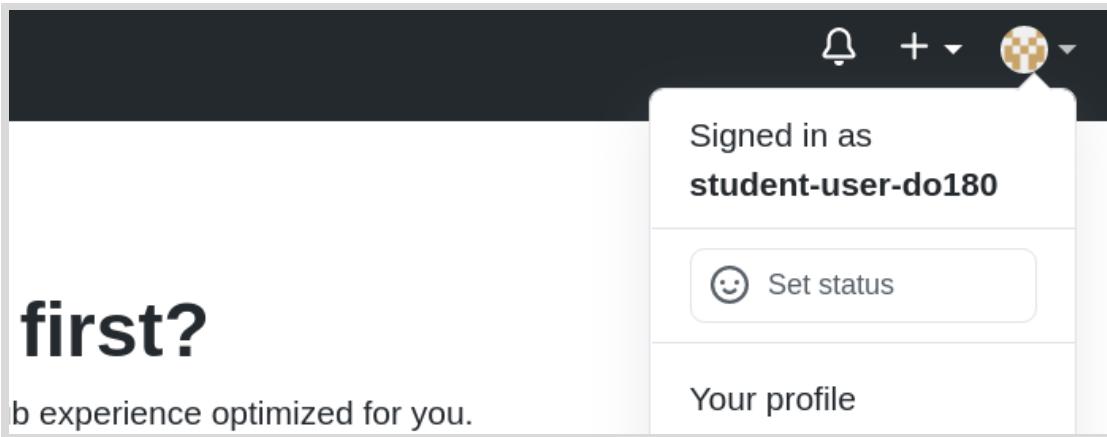


Figura 1.2: Menu do usuário

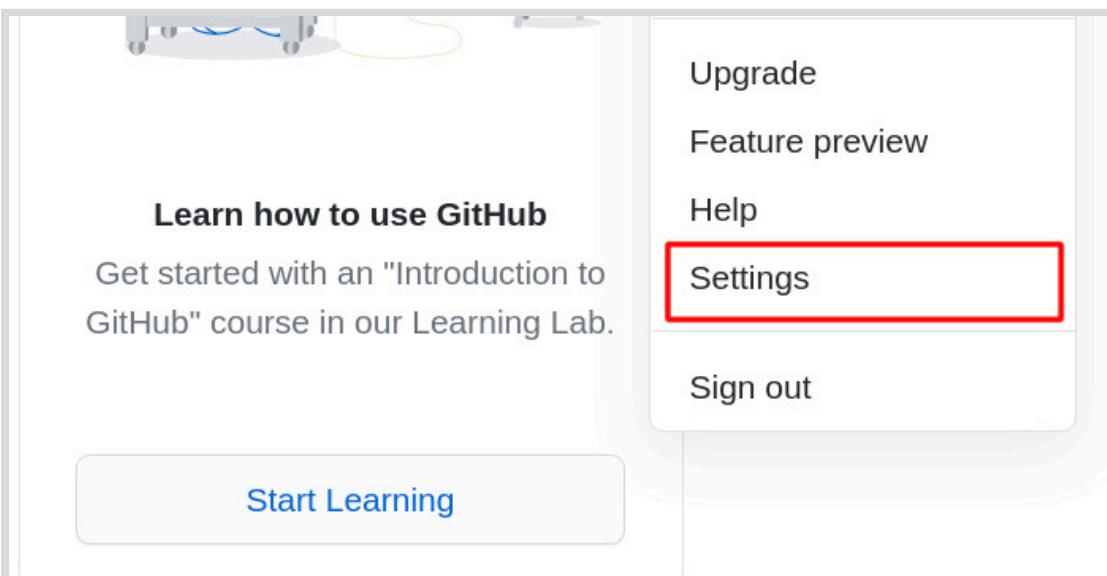


Figura 1.3: Menu de configurações

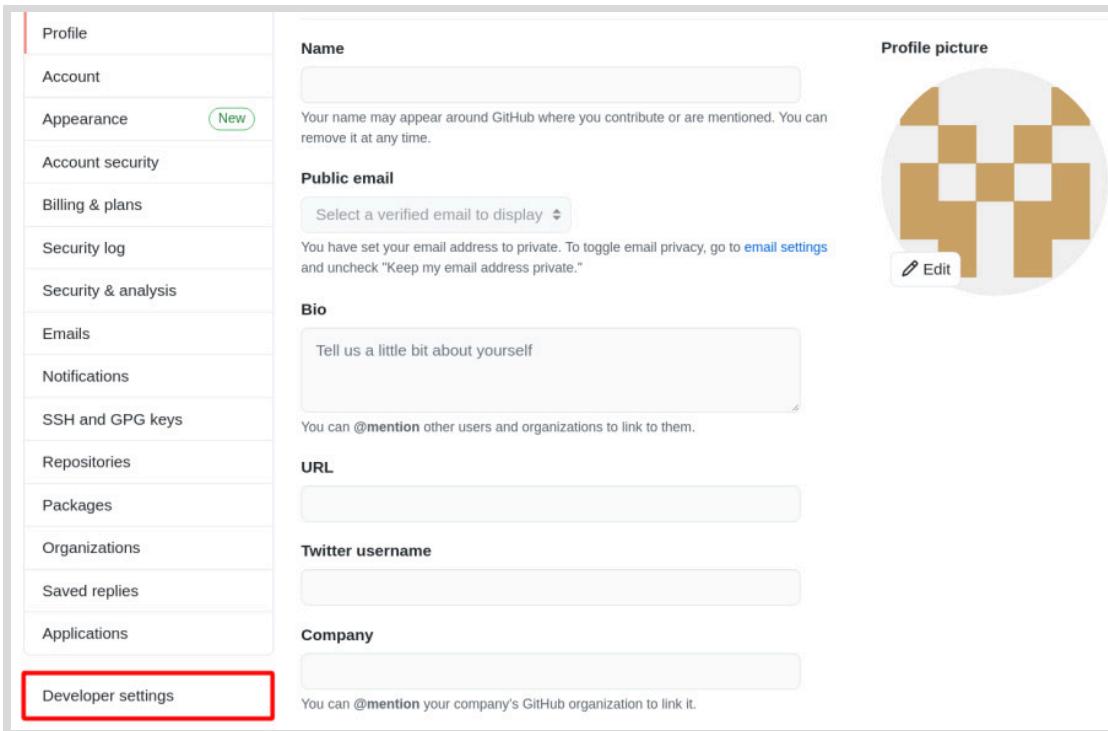


Figura 1.4: Configurações do desenvolvedor

- 1.3. Selecione a seção Personal access token no painel esquerdo. Na página seguinte, crie seu novo token clicando em **Generate new token**. Será solicitado que você digite sua senha.

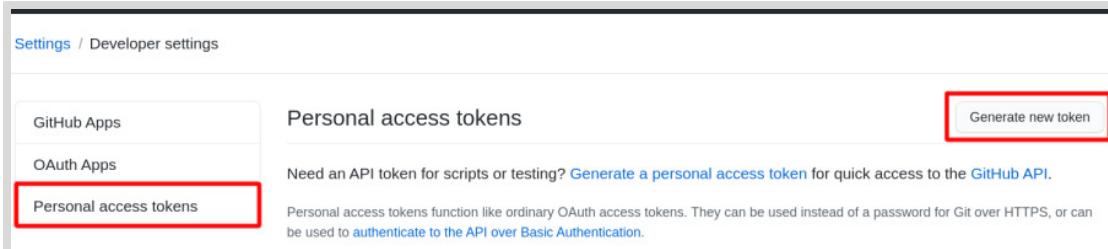


Figura 1.5: Painel de token de acesso pessoal

- 1.4. Escreva uma breve descrição sobre o novo token de acesso no campo Note.
- 1.5. Selecione a opção `public_repo` e deixe as outras opções desmarcadas. Crie seu novo token de acesso clicando em **Generate token**.

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Note

Course DO180
What's this token for?

Select scopes

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

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

Figura 1.6: Configuração de token de acesso pessoal

- 1.6. Seu novo token de acesso pessoal é exibido na saída. Usando o editor de texto de sua preferência, crie um novo arquivo no diretório pessoal do aluno chamado `token` e cole no token de acesso pessoal gerado. O token de acesso pessoal não pode ser exibido novamente no GitHub.

Personal access tokens

Generate new token Revoke all

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

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

✓ ghp_kgYGZwCGE1CrdovkzuzeLTWvYY6eBX2l0vck Copy	Delete
---	--------

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

Figura 1.7: Token de acesso gerado

- 1.7. Na workstation execute o comando `git config` com os parâmetros `credential.helper cache` para armazenar na memória de cache suas credenciais para uso futuro. O parâmetro `--global` aplica a configuração a todos os seus repositórios.

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

**Importante**

Durante este curso, se for solicitada uma senha ao usar operações Git na linha de comando, use seu token de acesso como senha.

► 2. Prepare sua senha do Quay.io

- 2.1. Configure uma senha para sua conta do Quay.io. Na página *Account Settings*, clique no link *Change password*. Consulte *Apêndice C, Criação de uma conta no Quay* para obter mais detalhes.

► 3. Configure a máquina workstation.

Para as etapas a seguir, use os valores que o ambiente Red Hat Online Training Learning fornecer a você ao provisionar seu ambiente de laboratório on-line:

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. A red box highlights the 'OpenShift Details' table, which contains the following information:

OpenShift Details		
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 with an 'active' status, an 'ACTION' dropdown menu, and a green 'OPEN CONSOLE' button. A red box also highlights the 'workstation' row.

Abra um terminal na sua máquina **workstation** e execute o seguinte comando: Responda aos prompts interativos antes de iniciar qualquer outro exercício neste curso.

Se cometer um erro, você pode interromper o comando a qualquer momento usando **Ctrl +C** e começando novamente.

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

- 3.1. O comando **lab-configure** começa exibindo uma série de prompts interativos e usa padrões sensatos quando eles estão disponíveis.

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

capítulo 1 | Introdução à tecnologia de contêiner

```

· Enter the Password: yourpassword 3
· Enter the GitHub Account Name: yourgituser 4
· Enter the Quay.io Account Name: yourquayuser 5

...output omitted...

```

- 1** A URL da API mestre do seu cluster do OpenShift. Digite o URL como uma única linha, sem espaços ou quebras de linha. O Treinamento Red Hat fornece essas informações a você quando você provisiona seu ambiente de laboratório. Você precisa dessas informações para fazer login no cluster e também implantar aplicativos em contêineres.
- 2** **3** Seu nome de usuário e sua senha de desenvolvedor do OpenShift. O Treinamento Red Hat fornece essas informações a você quando você provisiona seu ambiente de laboratório. Você deve usar esse nome de usuário e senha para fazer login no OpenShift. Você também usa seu nome de usuário como parte dos identificadores, como nomes de host de rota e nomes de projetos, a fim de evitar o conflito com identificadores de outros alunos que compartilham o mesmo cluster do OpenShift com você.
- 4** **5** Os nomes de suas contas pessoais do GitHub e do Quay.io. Você precisa de contas válidas e gratuitas nesses serviços on-line para realizar os exercícios do curso. Se você nunca usou nenhum desses serviços on-line, consulte Apêndice B, Criação de uma conta no GitHub e Apêndice C, Criação de uma conta no Quay para obter instruções sobre como se registrar.

- 3.2. O comando `lab-configure` exibe todas as informações que você digitou e tenta se conectar ao seu cluster do OpenShift.

```

...output omitted...

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

```

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

- 3.3. Se o `lab-configure` encontrar algum problema, ele exibirá uma mensagem de erro e será fechado. Será necessário verificar suas informações e executar o comando `lab-configure` novamente. A listagem a seguir mostra um exemplo de um erro de verificação.

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

Verifying your API Endpoint...

ERROR:

Cannot connect to an OpenShift 4 API using your URL.

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

capítulo 1 | Introdução à tecnologia de contêiner

- 3.4. Se tudo estiver OK até o momento, o `lab-configure` tentará acessar suas contas públicas do GitHub e do Quay.io.

```
...output omitted...

Verifying your GitHub account name...

Verifying your Quay.io account name...

...output omitted...
```

- 3.5. Se o `lab-configure` encontrar algum problema, ele exibirá uma mensagem de erro e será fechado. Será necessário verificar suas informações e executar o comando `lab-configure` novamente. A listagem a seguir mostra um exemplo de um erro de verificação:

```
...output omitted...

Verifying your GitHub account name...

ERROR:
Cannot find a GitHub account named: invalidusername.
```

- 3.6. Por fim, o comando `lab-configure` verifica se o seu cluster do OpenShift relata o domínio curinga esperados.

```
...output omitted...

Verifying your cluster configuration...

...output omitted...
```

- 3.7. Se todas as verificações forem aprovadas, o comando `lab-configure` salva sua configuração:

```
...output omitted...

Saving your lab configuration file...

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

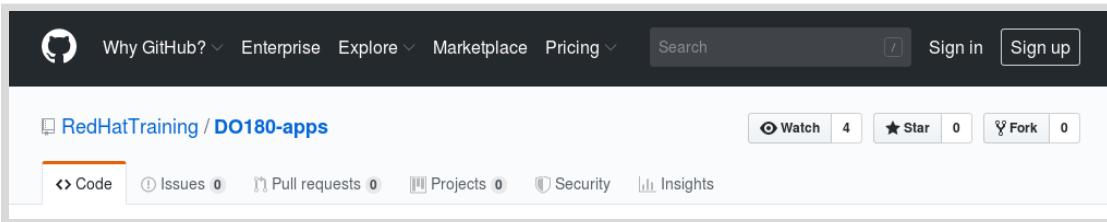
If you need to modify the configuration, rerun this or directly modify the values
in /usr/local/etc/ocp4.config.
```

- 3.8. Se não houver erros ao salvar sua configuração, você estará quase pronto para iniciar qualquer um dos exercícios deste curso. Se houver erros, não tente iniciar nenhum dos exercícios até que você consiga executar o comando `lab-configure` com êxito.

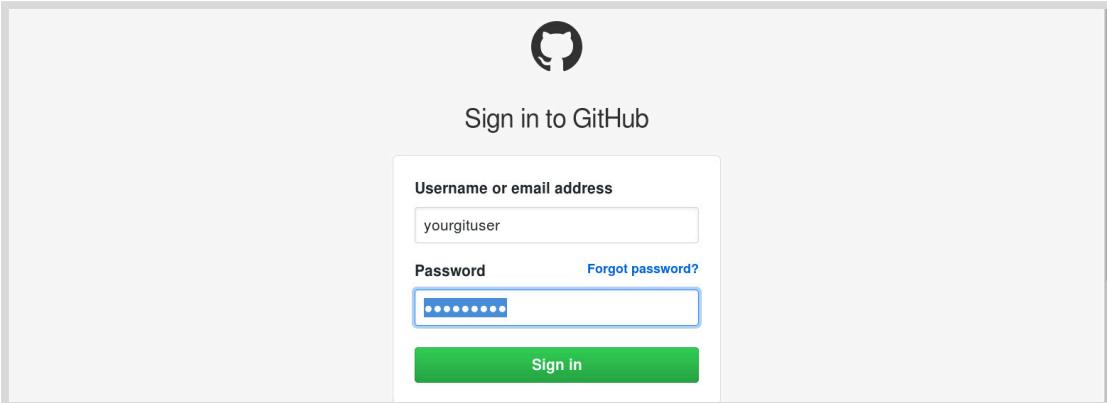
- 4. Ramifique os aplicativos de exemplo deste curso para sua conta pessoal do GitHub. Execute estas etapas:

capítulo 1 | Introdução à tecnologia de contêiner

- 4.1. Abra um navegador da web e acesse <https://github.com/RedHatTraining/DO180-apps>. Se você não estiver conectado ao GitHub, clique em **Sign in** no canto superior direito.



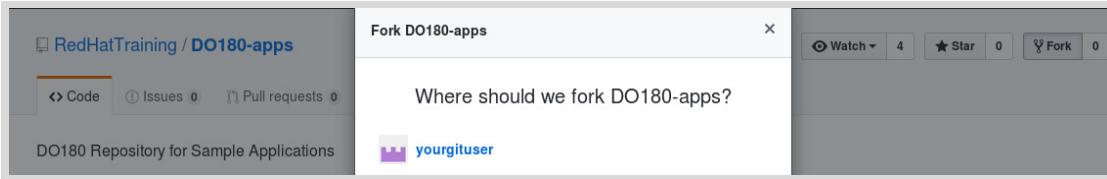
- 4.2. Faça login no GitHub usando seu nome de usuário pessoal e sua senha.



- 4.3. Navegue ao repositório **RedHatTraining/DO180-apps** e clique em **Fork** no canto superior direito.

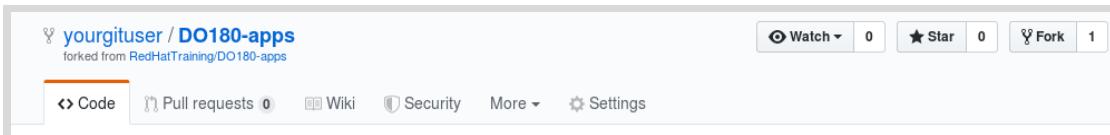


- 4.4. Na janela **Fork DO180-apps**, clique em `yourgituser` para selecionar seu projeto pessoal do GitHub.

**Importante**

Embora seja possível renomear sua ramificação pessoal do repositório <https://github.com/RedHatTraining/DO180-apps>, os scripts de classificação, os scripts auxiliares e a saída de exemplo neste curso pressupõem que você retém o nome **DO180-apps** ao ramificar o repositório.

- 4.5. Após alguns minutos, a interface web do GitHub exibe seu novo repositório `yourgituser/DO180-apps`.



- 5. Clone os aplicativos de exemplo deste curso de sua conta pessoal do GitHub para sua máquina **workstation**. Execute estas etapas:

- 5.1. Execute o comando a seguir para clonar o repositório de aplicativos de exemplo deste curso. Substitua *yourgituser* pelo nome de sua conta pessoal do GitHub.

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

- 5.2. Verifique se `/home/user/DO180-apps` é um repositório Git.

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

- 5.3. Crie um novo branch para testar seu novo token de acesso pessoal.

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

- 5.4. Faça uma alteração no arquivo `TEST` e, em seguida, confirme ela no Git.

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

- 5.5. Envie as alterações para o branch de teste recentemente criada.

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

- ① Digite seu nome de usuário do GitHub
- ② Insira seu token de acesso pessoal

- 5.6. Faça outra alteração em um arquivo de texto, confirme-a e a envie. Você verá que não foram solicitados o usuário e a senha. Isso acontece devido ao comando `git config` que você realizou na etapa 1.7.

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

- 5.7. Verifique se /home/user/D0180-apps contém os aplicativos de exemplo deste curso e altere novamente para a pasta inicial do usuário.

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

Agora que você tem um clone local do repositório D0180-apps na máquina workstation e executou o comando lab-configure com êxito, você está pronto para iniciar os exercícios do curso.

Durante este curso, todos os exercícios que criam aplicativos da fonte iniciam no branch master do repositório git D0180-apps. Os exercícios que fazem alterações no código-fonte exigem que você crie novas ramificações para hospedar suas alterações, de modo que a ramificação master sempre contenha um bom ponto de partida conhecido. Se, por algum motivo, você precisar pausar ou reiniciar um exercício e precisar salvar ou descartar as alterações feitas em seus branches do Git, consulte Apêndice E, Comandos git úteis.

Isso conclui o exercício orientado.

Sumário

Neste capítulo, você aprendeu que:

- Os contêineres são tempos de execução de aplicativo isolados criado com pouca sobrecarga.
- Uma imagem de contêiner empacota um aplicativo com todas as suas dependências, facilitando, assim, a execução do aplicativo em diferentes ambientes.
- Aplicativos como o Podman criam contêineres usando recursos do kernel padrão do Linux.
- Registros de imagem de contêiner são o mecanismo de preferência para distribuir imagens de contêiner para vários usuários e hosts.
- O OpenShift orquestra os aplicativos compostos por vários contêineres usando o Kubernetes.
- O Kubernetes gerencia o balanceamento de carga, a alta disponibilidade e o armazenamento persistente para aplicativos em contêineres.
- O OpenShift adiciona ao Kubernetes multilocação, segurança, facilidade de uso e recursos de integração e desenvolvimento contínuos.
- As rotas do OpenShift permitem acesso externo a aplicativos conteinerizados de maneira gerenciável.

capítulo 2

Criação de serviços conteinerizados

Meta

Provisionar um serviço usando tecnologia de contêineres.

Objetivos

- Criar um servidor de banco de dados a partir de uma imagem de contêiner.

Seções

- Provisionamento de um servidor de banco de dados conteinerizado (e exercício orientado)
- Uso dos contêineres sem raiz (e exercício orientado)

Laboratório

- Criação de serviços conteinerizados

Provisionamento de serviços conteinerizados

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Procurar e buscar imagens de contêiner com o Podman.
- Executar e configurar contêineres localmente.
- Usar o Red Hat Container Catalog.

Busca por imagens de contêiner com o Podman

Os aplicativos podem ser executados dentro de contêineres, fornecendo um ambiente de execução isolado e controlado. Executar um aplicativo conteinerizado, ou seja, executar um aplicativo dentro de um contêiner, exige uma imagem de contêiner, um pacote de sistema de arquivos que forneça todos os arquivos de aplicativos, bibliotecas e dependências que o aplicativo precisa executar. As imagens de contêiner estão disponíveis em registros de imagem que permitem aos usuários pesquisar e recuperar imagens de contêiner. Os usuários do Podman podem usar o subcomando `search` para encontrar imagens disponíveis em registros remotos ou locais.

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

Depois de encontrar uma imagem, você pode usar o Podman para baixá-la. Use o subcomando `pull` para direcionar o Podman a buscar a imagem e salvá-la localmente para uso futuro.

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

Imagens de contêiner são nomeadas com base na seguinte sintaxe:

```
registry_name/user_name/image_name:tag
```

Sintaxe de nomes de registro:

- `registry_name` é o nome do registry que armazena a imagem. Normalmente, é o FQDN do registro.
- `user_name` é o nome do usuário ou da organização à qual a imagem pertence.
- `image_name` deve ser exclusivo no namespace do usuário.
- A `tag` identifica a versão da imagem. Se o nome da imagem não incluir uma tag de imagem, pressupõe-se que a `latest` deve ser usada.

**nota**

A instalação do Podman da sala de aula usa vários registros disponíveis publicamente, como Quay.io e Red Hat Container Catalog.

Após a recuperação, o Podman armazena imagens localmente, e você pode listá-las com o subcomando `images`:

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

Execução dos contêineres

O comando `podman run` executa um contêiner localmente com base em uma imagem. No mínimo, o comando exige que o nome da imagem seja executado no contêiner.

A imagem do contêiner especifica um processo que inicia dentro do contêiner, conhecido como ponto de entrada. O comando `podman run` usa todos os parâmetros após o nome da imagem como o comando do ponto de entrada para o contêiner. O exemplo a seguir inicia um contêiner a partir de uma imagem do Red Hat Universal Base Image. Ele define o ponto de entrada desse contêiner para o comando `echo "Hello world"`:

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

Para iniciar uma imagem de contêiner como um processo em segundo plano, transmita a opção `-d` ao comando `podman run`:

```
[user@demo ~]$ podman run -d -p 8080 registry.redhat.io/rhel8/httpd-24
ff4ec6d74e9b2a7b55c49f138e56f8bc46fe2a09c23093664fea7febcb3dfa1b2
[user@demo ~]$ podman port -l
8080/tcp -> 0.0.0.0:44389
[user@demo ~]$ curl http://0.0.0.0:44389
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

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

Este exemplo executa um servidor HTTP Apache conteinerizado em segundo plano. Ele usa a opção `-p 8080` para vincular a porta do servidor HTTP a uma porta local. Depois, ele usa o

capítulo 2 | Criação de serviços conteinerizados

comando `podman port` para recuperar a porta local na qual o contêiner escuta. Por fim, ele usa essa porta para criar a URL de destino e buscar a página raiz do servidor Apache HTTP. Essa resposta prova que o contêiner ainda está funcionando após o comando `podman run`.

**nota**

A maioria dos subcomandos do Podman aceitam o sinalizador `-l` (`l` para o mais recente) como substituto para a ID do contêiner. Esse sinalizador aplica o comando ao contêiner usado mais recentemente em qualquer comando do Podman.

**nota**

Se a imagem a ser executada não estiver disponível localmente ao usar o comando `podman run`, o Podman usa automaticamente `pull` para fazer download da imagem.

Ao fazer referência ao contêiner, o Podman reconhece um contêiner com o nome do contêiner ou a ID do contêiner gerado. Use a opção `--name` para definir o nome do contêiner ao executar o contêiner com o Podman. Os nomes dos contêineres devem ser exclusivos. Se o comando `podman run` não incluir nenhum nome de contêiner, o Podman gerará um aleatório exclusivo para você.

Se as imagens exigirem que o usuário interaja com o console, o Podman poderá redirecionar fluxos de entrada e saída do contêiner para o console. O subcomando `run` exige os sinalizadores `-t` e `-i` (ou o sinalizador `-it`) para habilitar a interatividade.

**nota**

Muitos sinalizadores do Podman também têm uma forma longa alternativa; algumas delas são explicadas abaixo:

- `-t` é equivalente a `--tty`, o que significa que um `pseudo-tty` (pseudoterminal) será alocado para o contêiner.
- `-i` é o mesmo que `--interactive`. Quando usada, a entrada padrão é mantida aberta no contêiner.
- `-d`, ou sua forma longa, `--detach`, significa que o contêiner é executado em segundo plano (separado). O Podman então imprime a ID do contêiner.

Consulte a documentação do Podman para ver a lista completa de sinalizadores.

O exemplo a seguir inicia um terminal Bash *dentro* do contêiner e executa, de maneira interativa, alguns comandos nele:

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

Alguns contêineres precisam ou podem usar parâmetros externos fornecidos na inicialização. A abordagem mais comum para fornecer e consumir esses parâmetros é através de variáveis de ambiente. O Podman pode inserir variáveis de ambiente em contêineres na inicialização, adicionando o sinalizador `-e` ao subcomando `run`:

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

O exemplo anterior inicia um contêiner de imagens do UBI que imprime as duas variáveis de ambiente fornecidas como parâmetros.

Outro caso de uso para variáveis de ambiente é a configuração de credenciais em um servidor de banco de dados MySQL.

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

Uso do Red Hat Container Catalog

A Red Hat mantém seu repositório de imagens de contêiner precisamente ajustadas. Usar esse repositório oferece aos clientes uma camada de proteção e confiabilidade contra vulnerabilidades desconhecidas, que poderiam ser causadas por imagens não testadas. O comando `podman` padrão é compatível com o Red Hat Container Catalog. O Red Hat Container Catalog oferece uma interface fácil de usar para pesquisar e explorar imagens de contêiner do repositório da Red Hat.

O Container Catalog também serve como uma interface única, que oferece acesso a diferentes aspectos de todas as imagens de contêiner disponíveis no repositório. Ele é útil para determinar qual é a melhor das várias versões de imagem de contêiner usando as notas do índice de integridade. As notas do índice de integridade indicam o estado da imagem e se ela contém as atualizações de segurança mais recentes.

O Container Catalog também dá acesso a documentação sobre erratas para uma imagem. Ela descreve as correções mais recentes de bugs e as melhorias em cada atualização. Além disso, ela sugere a melhor técnica para extrair uma imagem em cada sistema operacional.

As imagens a seguir destacam alguns dos recursos do Red Hat Container Catalog.

The screenshot shows the Red Hat Container Catalog interface. At the top, there's a header with the title "Container images" and a sub-header stating "Container images offer lightweight and self-contained software to enable deployment at scale." Below the header, there's a search bar with the query "Apache httpd" and a "Search" button. To the right of the search bar, it says "1 - 5 of 5". On the left, there are filters for "Provider" (IBM, Red Hat, Inc.), "Category" (Database & Data Management, Programming Languages & Runtimes, Web Services), and "Product". The main area displays three container images from Red Hat:

- rhel8/httpd-24** (by Red Hat, Inc.)
- rhel8/httpd-24** (by Red Hat, Inc.)
- ibm/couchdb3** (by IBM)

Each card includes the provider logo, image name, version, author, description, and last update time. A "Have feedback?" button is located at the bottom right of the search results.

Figura 2.1: Página de pesquisa do Red Hat Container Catalog

Conforme mostrado na imagem anterior, pesquisar Apache `httpd` na caixa de pesquisa do Container Catalog exibe uma lista sugerida de produtos e repositórios de imagem que correspondem ao padrão da pesquisa. Para acessar a página de imagem Apache `httpd 2.4`, selecione `rhel8/httpd-24` na lista sugerida.

Depois que você selecionar a imagem desejada, a página subsequente mostrará informações adicionais sobre ela:

The screenshot shows the detailed view for the Apache `httpd 2.4` image. At the top, it says "Home > Software > Container images > Apache httpd 2.4". The image is labeled "Standalone Image" and has the name "Apache httpd 2.4". It shows the repository "rhe8/httpd-24" and the architecture "amd64". The tag dropdown is set to "latest". To the right, there's a "Provided by" section with the Red Hat logo. Below this, there are tabs for "Overview", "Security", "Packages", "Dockerfile", and "Get this image".

The "Description" section contains a detailed description of Apache HTTP Server 2.4. The "Published" section shows the date "28 days ago". The "Release category" is "Generally Available". The "Health index" shows a green bar with a value of "2". A "Have feedback?" button is located at the bottom right.

Figura 2.2: Página de imagem de visão geral Apache httpd 2.4 (`rhe8/httpd-24`)

O painel `Apache httpd 2.4` exibe detalhes da imagem e várias abas. Essa página declara que a Red Hat mantém o repositório de imagens.

capítulo 2 | Criação de serviços conteinerizados

Na guia Overview, há outros detalhes:

- *Description*: um sumário dos recursos da imagem.
- *Documentation*: referências sobre a documentação de autoria do contêiner.
- *Products using this container*: indica que o Red Hat Enterprise Linux usa esta imagem repositório.

No lado direito, ela mostra informações sobre quando a imagem recebeu a atualização mais recente, a tag mais recente aplicada à imagem, sua integridade, tamanho e outros dados.

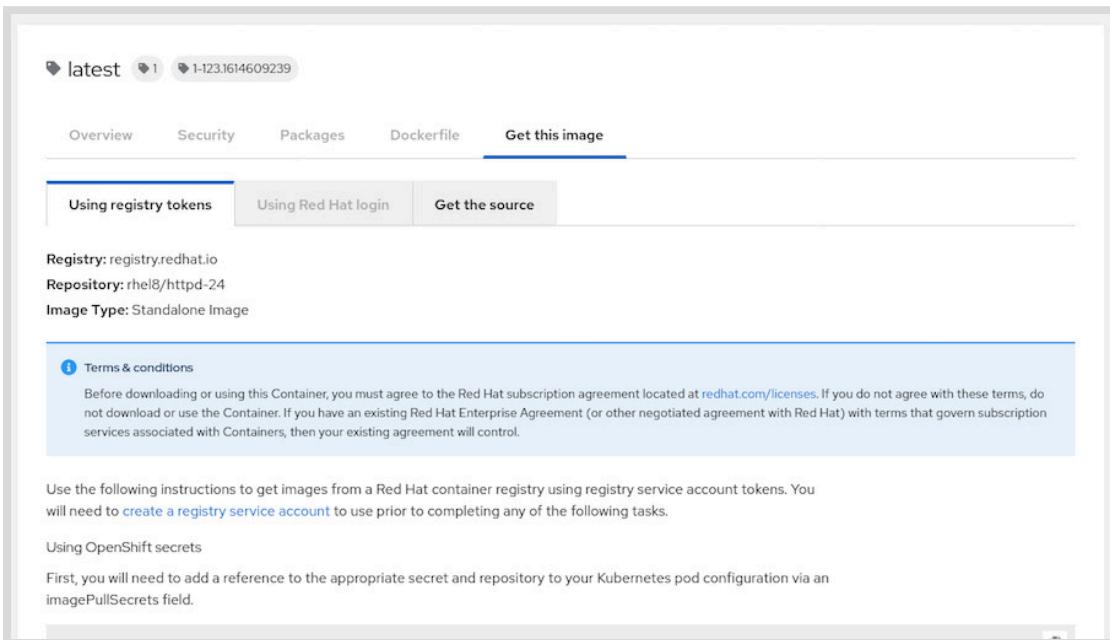


Figura 2.3: Página de imagem mais recente do Apache httpd 2.4 (rhel8/httpd-24)

A guia *Get this image* fornece o procedimento para obter a versão mais atual da imagem. A página fornece opções diferentes para recuperar a imagem. Escolha seu procedimento preferido nas guias, e a página fornece as instruções apropriadas para recuperar a imagem.

 **Referências**

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

Site do Quay.io
<https://quay.io>

► Exercício Guiado

Criação de uma instância de banco de dados do MySQL

Neste exercício, você iniciará um banco de dados do MySQL dentro de um contêiner e, em seguida, criará e preencherá um banco de dados.

Resultados

Você deverá ser capaz de iniciar um banco de dados a partir de uma imagem de contêiner e armazenar informações dentro do banco de dados.

Antes De Começar

Abra um terminal na workstation como usuário `student` e execute o seguinte comando:

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

Instruções

- 1. Crie uma instância de contêiner do MySQL.
 - 1.1. Faça login no Red Hat Container Catalog com sua conta da Red Hat. Se precisar se registrar na Red Hat, consulte as instruções em *Apêndice D, Criação de uma conta da Red Hat*.

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

- 1.2. Inicie um contêiner da imagem MySQL do Red Hat Container Catalog.

```
[student@workstation ~]$ podman run --name mysql-basic \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> -d registry.redhat.io/rhel8/mysql-80:1
Trying to pull ...output omitted...
Copying blob ...output omitted...
Writing manifest to image destination
Storing signatures
`2d37682eb33a`70330259d6798bdfdc37921367f56b9c2a97339d84faa3446a03
```

Esse comando faz download da imagem de contêiner do MySQL 8.0 com a tag 1 e inicia um contêiner baseado naquela imagem . Ele cria um banco de dados chamado `items` de propriedade do usuário chamado `user1` e com a senha `mypa55`. A senha de administrador do banco de dados é `r00tpa55`, e o contêiner é executado em segundo plano.

- 1.3. Verifique se o contêiner foi iniciado sem erros.

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

- 2. Acesse a área restrita do contêiner executando o comando a seguir:

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

O comando inicia um shell Bash, em execução como o usuário `mysql`, dentro do contêiner MySQL.

- 3. Adicione dados ao banco de dados.

- 3.1. Conecte-se ao MySQL com o usuário administrador do banco de dados (root).

Execute o comando a seguir no terminal do contêiner para se conectar ao banco de dados.

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

O comando `mysql` abre o prompt interativo do banco de dados MySQL. Execute o comando a seguir para determinar a disponibilidade do banco de dados.

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

- 3.2. Crie uma nova tabela no banco de dados `items`. Execute o seguinte comando para acessar o banco de dados.

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

- 3.3. Crie uma tabela chamada `Projects` no banco de dados `items`.

capítulo 2 | Criação de serviços conteinerizados

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

Você pode usar o arquivo ~/D0180/solutions/container-create/create_table.txt para copiar e colar a instrução CREATE TABLE do MySQL fornecida.

- 3.4. Use o comando `show tables` para verificar se a tabela foi criada.

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

- 3.5. Use o comando `insert` para inserir uma linha na tabela.

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

- 3.6. Use o comando `select` para verificar se as informações do projeto foram adicionadas à tabela.

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

- 3.7. Saia do prompt do MySQL e do contêiner do MySQL.

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

Encerramento

Na workstation, execute o script `lab container-create finish` para concluir esse laboratório.

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

Isso conclui o exercício.

Uso de contêineres sem raiz

Objetivos

Depois de concluir esta seção, você deverá ser capaz de:

- Explicar as diferenças entre executar contêineres raiz e sem raiz.
- Descrever as vantagens e desvantagens de cada caso.
- Executar como contêineres raiz e sem raiz com o Podman.

Evolução do uso de contêineres

Se você estiver executando contêineres há algum tempo, as chances são de que você os está executando como um usuário com privilégios. Historicamente, as ferramentas para criação de contêiner exigem que os mecanismos de tempo de execução sejam executados como root e o acesso privilegiado seja necessário para criar recursos, como interfaces de rede.

Do ponto de vista da segurança, fornecer esse nível de acesso é uma prática ruim. Você deve sempre executar software com privilégios o mais limitados possível. Quando um bug de segurança é explorado, no mecanismo de tempo de execução ou no próprio aplicativo, o impacto é minimizado.

Uma prática melhor é conteinerizar aplicativos para que sua execução não exija um usuário com privilégios. Em vez disso, esses aplicativos devem usar um usuário conhecido.

Muitas imagens de contêiner da comunidade, como as disponíveis em docker.io, ainda exigem o root para serem executadas.

Algumas ferramentas, como Podman e Red Hat OpenShift, executam contêineres sem raiz por padrão. O Docker anunciou que o modo sem raiz está disponível na versão 20.10.

Vantagens dos contêineres sem raiz

Contêineres sem raiz são um novo conceito de contêineres que não exigem privilégios de root para ser executados. Os contêineres sem raiz são vantajosos do ponto de vista da segurança por vários motivos, incluindo:

- Permitem que o código seja executado dentro de um contêiner sem raiz com privilégios de root, sem precisar ser executado como o usuário root do host.
- Adicionam uma nova camada de segurança; se o mecanismo de contêiner for comprometido, o invasor não obterá privilégios de root no host.
- Permitem que vários usuários sem direitos executem contêineres na mesma máquina.
- Permitem o isolamento dentro de contêineres aninhados.

Apesar de todas essas vantagens, a execução de contêineres sem raiz tem várias limitações, incluindo:

- Recursos descartados

capítulo 2 | Criação de serviços conteinerizados

- Vinculação a portas com menos de 1024
- Montagem de volume de outro conteúdo

Há uma lista detalhada de deficiências dessa método em <https://github.com/containers/podman/blob/master/rootless.md>.

Compreensão de contêineres sem raiz

Para entender como os contêineres sem raiz funcionam, considere os seguintes conceitos.

Namespace de usuário

Os contêineres usam namespaces do Linux para isolarem-se do host no qual são executados. O namespace de usuário é usado principalmente para tornar os contêineres sem raiz. Esse namespace mapeia as IDs de usuário e grupo para que um processo dentro do namespace pareça estar em execução em uma ID diferente.

Os contêineres sem raiz usam o namespace de usuário para fazer com que o código do aplicativo pareça estar sendo executado como root. No entanto, da perspectiva do host, as permissões são limitadas às de um usuário regular. Se um invasor conseguir escapar do namespace de usuário no host, ele terá apenas os recursos de um usuário normal e sem direitos.

Rede

Para permitir uma rede adequada dentro de um contêiner, é criado um dispositivo Ethernet virtual. Isso representa um problema para contêineres sem raiz, pois somente um usuário root real tem privilégios para criar esse e dispositivos semelhantes.

Em um contêiner sem raiz, a rede geralmente é gerenciada pelo Slirp. Ele funciona fazendo uma ramificação nos namespaces de usuário e rede e criando um dispositivo TAP que se torna a rota padrão. Em seguida, ele passa o descritor de arquivo do dispositivo para o pai, que é executado no namespace de rede padrão e agora pode se comunicar com o contêiner e com a internet.

Armazenamento

Por padrão, os mecanismos de contêiner usam um driver especial chamado Overlay2 (ou Overlay) para criar um sistema de arquivos em camadas que seja eficiente na capacidade e no desempenho. Isso não pode ser feito com contêineres sem raiz, pois a maioria das distribuições do Linux não permite a montagem de sistemas de arquivos de sobreposição em namespaces de usuário.

Para contêineres sem raiz, a solução é criar um novo driver de armazenamento. O FUSE-OverlayFS é uma implementação de espaço de usuário do Overlay, que é mais eficiente do que o driver de armazenamento VFS usado antes e pode ser executado dentro de namespaces de usuário.



Referências

página do man user_namespaces(7)

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

Uso do Podman em um ambiente sem raiz

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

► Exercício Guiado

Exploração de contêineres raiz e sem raiz

Neste exercício, você entenderá as diferenças entre executar contêineres como root e executá-los sem raiz.

Resultados

Você deverá ser capaz de ver as UIDs dos processos em execução em contêineres.

Antes De Começar

Com o usuário `student` na máquina `workstation`, use o comando `lab` para preparar seu sistema para este exercício.

Esse comando garante que você tenha disponível as ferramentas necessárias para realizar este exercício.

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

Instruções

- 1. Execute um contêiner como o usuário root e verifique a UID de um processo em execução nele.

- 1.1. Inicie um contêiner com sudo a partir da imagem Red Hat UBI 8.

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

- 1.2. Inicie um processo `sleep` no contêiner.

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

- 1.3. Em um novo terminal, execute `ps` para pesquisar o processo.

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

- 1.4. Saia do contêiner.

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

- 2. Execute um contêiner como o usuário comum e verifique a UID de um processo em execução nele.

2.1. Inicie outro contêiner da Red Hat UBI 8 como um usuário comum.

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

2.2. Inicie um processo `sleep` no contêiner.

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

2.3. Em um novo terminal, execute `ps` para pesquisar o processo.

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

2.4. Saia do contêiner.

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

Encerramento

Na máquina `workstation`, use o comando `lab` para concluir este exercício. Esse passo é importante para garantir que recursos de exercícios anteriores não afetem exercícios futuros.

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

Isso conclui a seção.

► Laboratório Aberto

Criação de serviços conteinerizados

Resultados

Você deverá ser capaz de iniciar e personalizar um contêiner usando uma imagem de contêiner.

Antes De Começar

Abra um terminal na máquina `workstation` como usuário `student` e execute o seguinte comando:

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

Instruções

1. Use a imagem `quay.io/redhattraining/httpd-parent` com a tag `2.4` para iniciar um novo contêiner chamado `httpd-basic` em segundo plano. Encaminhe a porta `8080` no host para a porta `80` no contêiner. Use a opção `-p` do comando `podman` e defina o valor da opção como `8080:80`.
2. Teste se o servidor Apache HTTP está sendo executado no contêiner `httpd-basic`.
3. Personalize o contêiner `httpd-basic` para exibir `Hello World` como mensagem. A mensagem do contêiner é armazenada no arquivo `/var/www/html/index.html`.

Avaliação

Classifique seu trabalho executando o comando `lab container-review grade` na sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na máquina `workstation`, execute o script `lab container-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

► Solução

Criação de serviços conteinerizados

Resultados

Você deverá ser capaz de iniciar e personalizar um contêiner usando uma imagem de contêiner.

Antes De Começar

Abra um terminal na máquina `workstation` como usuário `student` e execute o seguinte comando:

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

Instruções

1. Use a imagem `quay.io/redhattraining/httpd-parent` com a tag `2.4` para iniciar um novo contêiner chamado `httpd-basic` em segundo plano. Encaminhe a porta `8080` no host para a porta `80` no contêiner. Use a opção `-p` do comando `podman` e defina o valor da opção como `8080:80`.
 - 1.1. Use o comando `podman run` para iniciar um contêiner. Adicione a opção `-d` para iniciá-lo em segundo plano e adicione a opção `-p` para mapear a porta `8080` no host para a porta `80` no contêiner.

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

Esse comando inicia o Apache HTTP Server em segundo plano e retorna ao prompt do Bash.

2. Teste se o servidor Apache HTTP está sendo executado no contêiner `httpd-basic`.
 - 2.1. Na máquina `workstation`, tente acessar `http://localhost:8080` usando qualquer navegador.

Uma mensagem `Hello from the httpd-parent container!` é exibida. Essa mensagem está localizada na página `index.html` do contêiner do servidor Apache HTTP em execução na máquina `workstation`.

capítulo 2 | Criação de serviços conteinerizados

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

3. Personalize o contêiner `httpd-basic` para exibir Hello World como mensagem. A mensagem do contêiner é armazenada no arquivo `/var/www/html/index.html`.

- 3.1. Inicie uma sessão Bash dentro do contêiner.

Executar o seguinte comando:

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

- 3.2. Na sessão Bash, verifique o arquivo `index.html` no diretório `/var/www/html` usando o comando `ls -la`.

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

- 3.3. Altere o arquivo `index.html` para conter o texto Hello World, substituindo o conteúdo existente.

Na sessão Bash no contêiner, execute o seguinte comando:

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

- 3.4. Tente novamente acessar `http://localhost:8080` e verifique se a página da web foi atualizada.

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

Avaliação

Classifique seu trabalho executando o comando `lab container-review grade` na sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na máquina `workstation`, execute o script `lab container-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

Sumário

Neste capítulo, você aprendeu que:

- O Podman permite que os usuários procurem e baixem imagens de registros locais ou remotos.
- O comando `podman run` cria um novo contêiner e inicia um contêiner a partir de uma imagem de contêiner.
- Os contêineres são executados em segundo plano usando o sinalizador `-d`, ou interativamente, usando o sinalizador `#it`.
- Algumas imagens de contêiner exigem variáveis de ambiente que são definidas usando a opção `-e` no comando `podman run`.
- O Red Hat Container auxilia a pesquisar, explorar e analisar imagens de contêiner do repositório oficial de imagens de contêiner da Red Hat.

capítulo 3

Gerenciamento de contêineres

Meta

Usar imagens predefinidas de contêineres para criar e gerenciar serviços conteinerizados.

Objetivos

- Gerenciar o ciclo de vida de um contêiner da criação à exclusão.
- Salvar dados do aplicativo de contêiner com armazenamento persistente.
- Descrever como usar o encaminhamento de portas para acessar um contêiner.

Seções

- Gerenciamento do ciclo de vida de contêineres (e exercício orientado)
- Anexação de armazenamento persistente aos contêineres (e exercício orientado)
- Acesso aos contêineres (e exercício orientado)

Laboratório

- Gerenciamento de contêineres

Gerenciamento do ciclo de vida de contêineres

Objetivos

Após a conclusão desta seção, os alunos deverão ser capazes de gerenciar o ciclo de vida de um contêiner da criação à exclusão.

Gerenciamento do ciclo de vida do contêiner com Podman

Nos capítulos anteriores, você aprendeu a usar o Podman para criar um serviço conteinerizado. Agora você vai mergulhar mais fundo em comandos e estratégias que você pode usar para gerenciar o ciclo de vida de um contêiner. O Podman não só permite que você execute contêineres, mas também que os execute em segundo plano, que execute novos processos dentro deles e que forneça recursos, como volumes do sistema de arquivos ou uma rede.

O Podman, implementado pelo comando `podman`, fornece um conjunto de subcomandos para criar e gerenciar contêineres. Os desenvolvedores usam esses subcomandos para gerenciar o contêiner e o ciclo de vida da imagem do contêiner. A figura a seguir exibe um resumo dos subcomandos mais usados para alterar o estado do contêiner e da imagem:

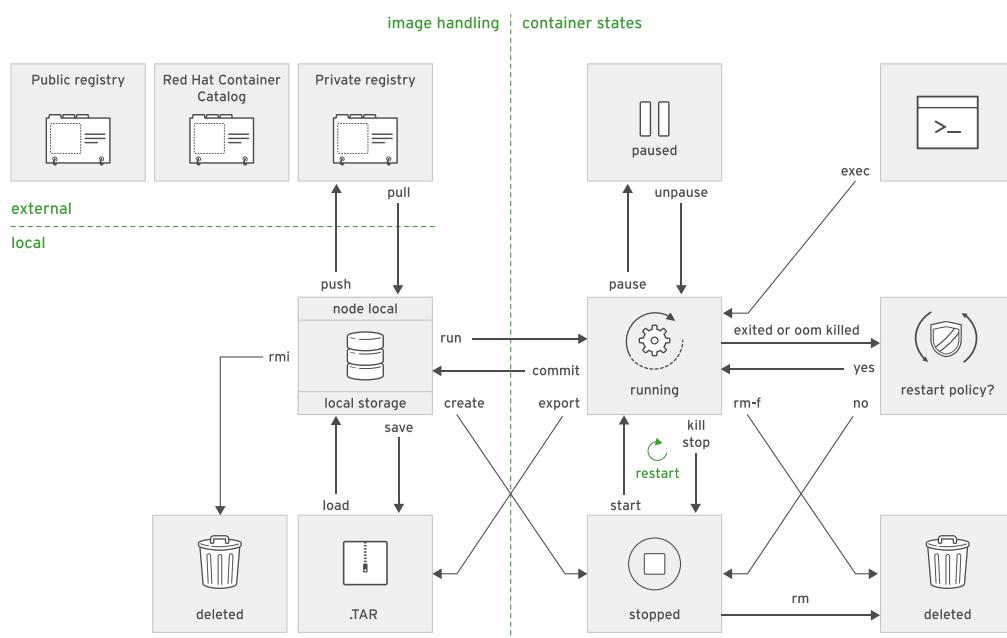


Figura 3.1: Gerenciamento de subcomandos do Podman

O Podman também fornece um conjunto útil de subcomandos para obter informações sobre contêineres em execução e interrompidos.

Você pode usar esses subcomandos para extrair informações de contêineres e imagens para fins de depuração, atualização ou relatórios. A figura a seguir exibe um resumo dos subcomandos mais usados que consultam informações dos contêineres e das imagens:

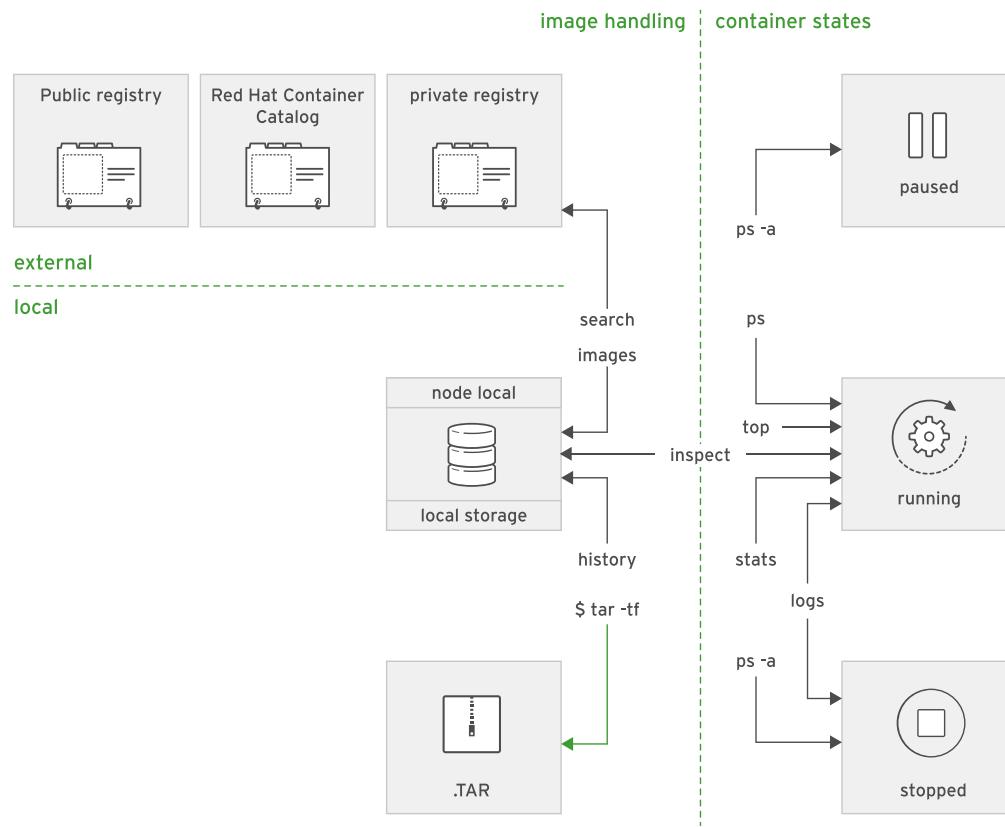


Figura 3.2: Subcomandos de consulta do Podman

Use essas duas figuras como uma referência enquanto aprende sobre os subcomandos do Podman neste curso.

Criação de contêineres

O comando `podman run` cria um novo contêiner a partir de uma imagem e inicia um processo no novo contêiner. Se a imagem do contêiner não estiver disponível localmente, este comando tentará fazer download da imagem usando o repositório de imagens configurado:

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

Nesta amostra de saída, o contêiner foi iniciado com um processo não interativo (sem a opção `-it`) e está sendo executado em primeiro plano porque não foi iniciado com a opção `-d`. Portanto, interromper o processo resultante com `Ctrl+C` (`SIGINT`) interrompe o processo do contêiner e o próprio contêiner.

O Podman identifica contêineres com uma ID de contêiner exclusiva ou nome de contêiner. O comando `podman ps` exibe a ID do contêiner e os nomes de todos os contêineres ativamente em execução:

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

- ① A ID do contêiner é exclusiva e gerada automaticamente.
- ② O nome do contêiner pode ser especificado manualmente, caso contrário, ele será gerado automaticamente. Esse nome deve ser exclusivo; se não for, o comando run falhará.

O comando `podman run` gera automaticamente uma ID exclusiva e aleatória. Ele também gera um nome de contêiner aleatório. Para definir explicitamente o nome do contêiner, use a opção `--name` ao executar um contêiner:

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



nota

O nome deve ser único. O Podman apresenta um erro se o nome já estiver em uso por qualquer contêiner, incluindo contêineres interrompidos.

Outro recurso importante é a capacidade de executar o contêiner como um processo de daemon em segundo plano. A opção `-d` é responsável pela execução no modo desconectado. Ao usar essa opção, o Podman retorna a ID do contêiner na tela, permitindo que você continue executando comandos no mesmo terminal enquanto o contêiner é executado em segundo plano:

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

A imagem do contêiner especifica que o comando seja executado para iniciar o processo conteinerizado, conhecido como o ponto de entrada. O comando `podman run` pode substituir esse ponto de entrada incluindo o comando após a imagem do contêiner:

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

O comando especificado deve ser executável dentro da imagem de contêiner.



nota

Como um comando especificado aparece no exemplo, o contêiner ignora o ponto de entrada para a imagem `httpd`. Portanto, o serviço `httpd` não é iniciado.

Alguns contêineres precisam ser executados como um processo ou shell ou interativo. Isso inclui contêineres que executam processos que precisam de entrada do usuário (como inserir comandos), e processos que geram saída através da saída padrão. O seguinte exemplo inicia um shell bash interativo em um contêiner `registry.redhat.io/rhel8/httpd-24`:

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

As opções `-t` e `-i` permitem redirecionamento de terminal para programas interativos baseados em texto. A opção `-t` aloca um pseudo-tty (um terminal) e o anexa à entrada padrão do contêiner. A opção `-i` mantém a entrada padrão do contêiner aberta, mesmo que tenha sido desconectada, para que o processo principal possa continuar aguardando a entrada.

Execução de comandos em um contêiner

Quando um contêiner é iniciado, ele executa o comando a partir do ponto de entrada. Porém, pode ser necessário executar outros comandos para gerenciar o contêiner em execução.

Alguns casos de uso típicos são mostrados abaixo:

- Execução de um shell interativo em um contêiner já em execução.
- Processos em execução que atualizam ou exibem os arquivos do contêiner.
- Inicialização de novos processos em segundo plano dentro do contêiner.

O comando `podman exec` inicia um processo adicional dentro de um contêiner já execução:

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

Neste exemplo, a ID do contêiner é usada para executar um comando em um contêiner existente.

O Podman lembra do último contêiner criado. Os desenvolvedores podem ignorar a gravação da ID ou do nome desse contêiner nos comandos posteriores do Podman, substituindo a ID do contêiner pela opção `-l` (ou `--latest`):

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

Gerenciamento de contêineres

Criar e iniciar um contêiner é só a primeira etapa do ciclo de vida do contêiner. Esse ciclo de vida também inclui interromper, reiniciar ou remover o contêiner. Os usuários também podem examinar o status do contêiner e os metadados para fins de depuração, atualização ou geração de relatórios.

O Podman fornece os seguintes comandos para gerenciar contêineres:

- `podman ps`: esse comando lista os contêineres em execução:

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

- ❶ Cada linha descreve informações sobre o contêiner.

capítulo 3 | Gerenciamento de contêineres

Cada contêiner, quando criado, recebe uma **container ID**, que é um número hexadecimal. Essa ID parece uma ID de imagem, mas não é relacionada.

O campo **IMAGE** indica a imagem de contêiner que era usada para iniciar o contêiner.

O campo **COMMAND** indica o comando executado quando o contêiner foi iniciado.

O campo **CREATED** indica a data e hora em que o contêiner foi iniciado.

O campo **STATUS** indica o tempo de atividade total do contêiner, se ainda estiver em execução, ou tempo desde o encerramento.

O campo **PORTS** indica portas que eram expostas pelo contêiner ou qualquer porta de encaminhamento que possa estar configurada.

O campo **NAMES** indica o nome do contêiner.

O Podman não descarta os contêineres interrompidos imediatamente. O Podman preserva seus sistemas de arquivos locais e outros estados para facilitar a análise *postmortem*. A opção **-a** lista todos os contêineres, incluindo os interrompidos:

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

**nota**

Ao criar contêineres, o Podman aborta se o nome do contêiner já estiver em uso, mesmo se o contêiner estiver com o status stopped (interrompido). Essa opção ajuda a evitar nomes de contêiner duplicados.

- **podman stop**: esse comando interrompe um contêiner normalmente:

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

Usar **podman stop** é mais fácil do que encontrar o processo de inicialização do contêiner no sistema operacional do host e encerrá-lo.

- **podman kill**: esse comando envia sinais Unix ao processo principal no contêiner. Se nenhum sinal for especificado, ele envia o sinal **SIGKILL**, encerrando o processo principal e o contêiner.

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

Você pode especificar o sinal com a opção **-s**:

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

Qualquer sinal Unix pode ser enviado ao processo principal. O Podman aceita o nome ou o número do sinal.

A tabela a seguir mostra vários sinais úteis:

Sinal	Valor	Ação padrão	Comentário
SIGHUP	1	Term	Desligamento detectado no terminal de controle ou encerramento do processo de controle
SIGINT	2	Term	Interromper a partir do teclado
SIGQUIT	3	Core	Sair partir do teclado
SIGILL	4	Core	Instrução ilegal
SIGABRT	6	Core	Anular sinal a partir de abort(3)
SIGFPE	8	Core	Exceção de ponto flutuante
SIGKILL	9	Term	Sinal de encerramento
SIGSEGV	11	Core	Referência de memória inválida
SIGPIPE	13	Term	Pipe quebrado: grave no pipe sem leitores
SIGALRM	14	Term	Sinal do temporizador a partir de alarm(2)
SIGTERM	15	Term	Sinal de encerramento
SIGUSR1	30,10,16	Term	Sinal 1 definido pelo usuário
SIGUSR2	31,12,17	Term	Sinal 2 definido pelo usuário
SIGCHLD	20,17,18	Ign	Filho interrompido ou encerrado
SIGCONT	19,18,25	Cont	Continuar, se interrompido
SIGSTOP	17,19,23	Stop	Interromper processo
SIGTSTP	18,20,24	Stop	Interromper o que foi digitado em tty
SIGTTIN	21,21,26	Stop	Entrada de tty para processo em segundo plano
SIGTTOU	22,22,27	Stop	Saída de tty para processo em segundo plano

Qualquer sinal Unix pode ser enviado ao processo principal. O Podman aceita o nome ou o número do sinal.

**nota****Termo**

Encerrar o processo.

Core

Encerrar o processo e gerar um despejo de memória.

Ign

O sinal é ignorado.

Stop

Interromper processo.

Outros comandos úteis do podman incluem:

- **podman restart**: este comando reinicia um contêiner interrompido.

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

O comando **podman restart** cria um novo contêiner com a mesma ID de contêiner, reutilizando o estado e o sistema de arquivos do contêiner interrompido.

- O comando **podman rm** exclui um contêiner e descarta seu estado e sistema de arquivos.

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

A opção **-f** do subcomando **rm** instrui o Podman a remover o contêiner mesmo que ele não esteja interrompido. Essa opção encerra o contêiner de maneira forçada e, depois, o remove. Usar a opção **-f** é equivalente aos comandos **podman kill** e **podman rm** juntos.

Você pode excluir todos os contêineres ao mesmo tempo. Muitos subcomandos **podman** aceitam a opção **-a**. Essa opção indica o uso do subcomando em todos os contêineres ou imagens disponíveis. O seguinte exemplo remove todos os contêineres:

```
[user@host ~]$ podman rm -a
5fd8e98ec7eab567eabe84943fe82e99fdfc91d12c65d99ec760d5a55b8470d6
716fd687f65b0957edac73b84b3253760e915166d3bc620c4aec8e5f4eadfe8e
86162c906b44f4cb63ba2e3386554030dcbaedbce9e9fcad60aa9f8b2d5d4
```

Antes de excluir todos os contêineres, todos os contêineres em execução devem ter o status **stopped** (interrompido). Você pode usar o seguinte comando para interromper todos os contêineres:

```
[user@host ~]$ podman stop -a
5fd8e98ec7eab567eabe84943fe82e99fdfc91d12c65d99ec760d5a55b8470d6
716fd687f65b0957edac73b84b3253760e915166d3bc620c4aec8e5f4eadfe8e
86162c906b44f4cb63ba2e3386554030dcbaedbce9e9fcad60aa9f8b2d5d4
```



nota

Os subcomandos `inspect`, `stop`, `kill`, `restart` e `rm` podem usar a ID do contêiner em vez do nome do contêiner.



Referências

Página do man de sinais do Unix Posix

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

► Exercício Guiado

Gerenciamento de um contêiner MySQL

Neste exercício, você criará e gerenciará um contêiner de banco de dados MySQL®.

Resultados

Você deverá ser capaz de criar e gerenciar um contêiner de banco de dados MySQL.

Antes De Começar

Certifique-se de que a máquina workstation tem o comando podman disponível e está corretamente configurada executando o seguinte comando a partir de uma janela de terminal:

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

Instruções

- 1. Faça download da imagem de contêiner do banco de dados MySQL e tente iniciá-la. O contêiner não é iniciado porque diversas variáveis de ambiente devem ser fornecidas à imagem.
 - 1.1. Faça login no Red Hat Container Catalog com sua conta da Red Hat. Se precisar se registrar na Red Hat, consulte as instruções em *Apêndice D, Criação de uma conta da Red Hat*.

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

- 1.2. Faça download da imagem de contêiner do banco de dados MySQL e tente iniciá-la.

```
[student@workstation ~]$ podman run --name mysql-db \
> registry.redhat.io/rhel8/mysql-80:1
Trying to pull ...output omitted...
...output omitted...
Writing manifest to image destination
Storing signatures
You must either specify the following environment variables:
  MYSQL_USER (regex: '^$')
  MYSQL_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=;<,.?;:|]$')
  MYSQL_DATABASE (regex: '^$')
`Or the following environment variable:
  MYSQL_ROOT_PASSWORD (regex: '^[a-zA-Z0-9_~!@#$%^&*()-=;<,.?;:|]$')
Or both.
Optional Settings:
```

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

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

**nota**

Se você tentar executar o contêiner como um daemon (-d), a mensagem de erro sobre as variáveis exigidas não será exibida. Porém, essa mensagem de erro é incluída como parte dos registros de contêiner, que podem ser exibidos usando o seguinte comando:

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

- 2. Crie um novo contêiner chamado `mysql` e especifique cada variável necessária usando o parâmetro `-e`.

**nota**

Certifique-se de iniciar o novo contêiner com o nome correto.

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

O comando exibe a ID do contêiner `mysql`. Veja abaixo um exemplo da saída.

```
a49dba9ff17f2b5876001725b581fdd331c9ab8b9eda21cc2a2899c23f078509
```

- 3. Verifique se o contêiner `mysql` foi iniciado corretamente. Executar o seguinte comando:

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

O comando mostra apenas os 12 primeiros caracteres da ID do contêiner exibidos no comando anterior.

- 4. Preencha o banco de dados `items` com a tabela `Projects`:

- 4.1. Execute o comando `podman cp` para copiar o arquivo de banco de dados para o contêiner `mysql`.

```
[student@workstation ~]$ podman cp \
> /home/student/D0180/labs/manage-lifecycle/db.sql mysql:/
```

- 4.2. Preencha o banco de dados `items` com a tabela `Projects`.

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

- 5. Crie outro contêiner usando a mesma imagem de contêiner anterior. Insira interativamente o shell /bin/bash em vez de usar o comando padrão para a imagem de contêiner.

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

- 6. Tente se conectar ao banco de dados MySQL no novo contêiner:

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

O seguinte erro é exibido:

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

O servidor de banco de dados MySQL não está em execução porque o contêiner executou o comando /bin/bash em vez de iniciar o servidor MySQL.

- 7. Saia do contêiner.

```
bash-4.4$ exit
```

- 8. Verifique se o contêiner mysql-2 não está em execução.

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

- 9. Consulte o contêiner mysql para listar todas as linhas na tabela Projects. O comando instrui o shell bash a consultar o banco de dados items usando um comando mysql.

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

Encerramento

Na workstation, execute o script a seguir para concluir este exercício.

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

Isso conclui o exercício.

Anexação de armazenamento persistente aos contêineres

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Salvar dados de aplicativos em remoções de contêineres com o uso de armazenamento persistente.
- Configurar diretórios de host para uso como volumes de contêineres.
- Montar um volume em um contêiner.

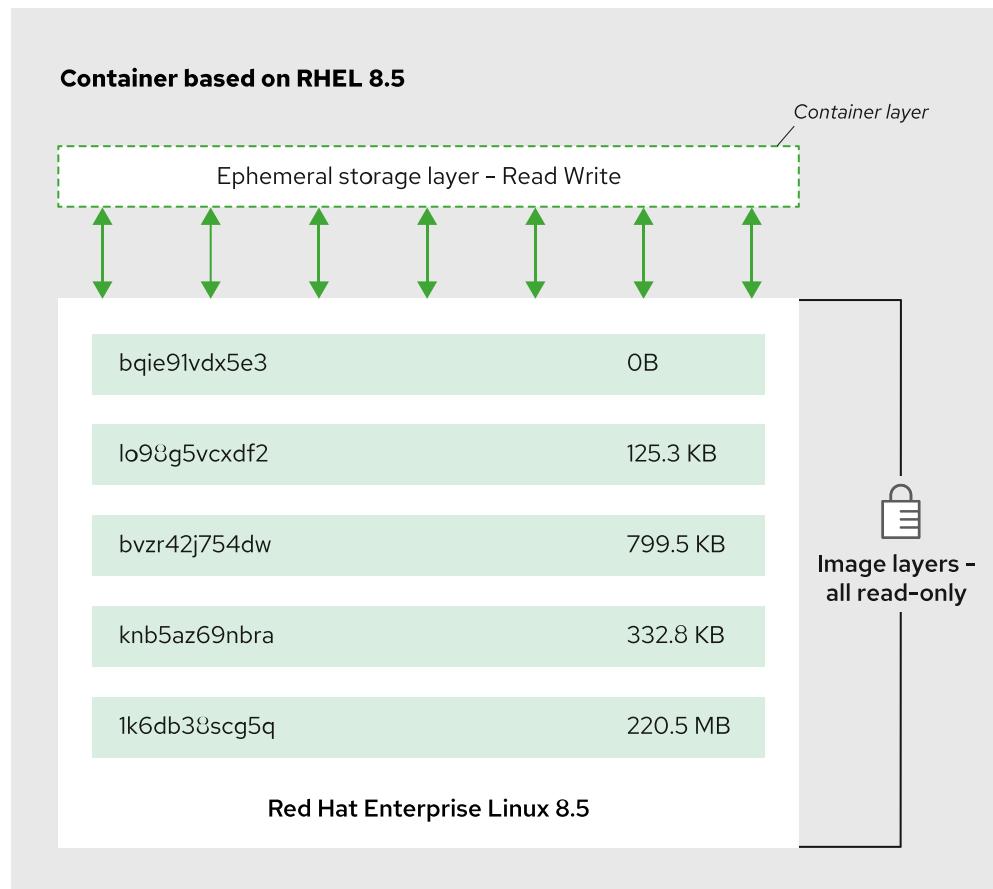
Preparação de locais permanentes de armazenamento

Dizem que o armazenamento de contêineres é *efêmero*, o que significa que o conteúdo não é preservado após o contêiner ser removido. Os aplicativos conteinerizados funcionam com base no pressuposto de que sempre são iniciados com o armazenamento vazio. Por isso, criar e excluir contêineres são operações relativamente baratas.

Anteriormente neste curso, as imagens de contêineres eram caracterizadas como *imutáveis* e *em camadas*, o que significa que nunca são alterados, mas compostos por camadas que adicionam ou substituem o conteúdo das camadas abaixo.

Um contêiner em execução obtém uma nova camada sobre sua imagem de contêiner de base, e essa camada é o *armazenamento de contêineres*. A princípio, essa camada é o único armazenamento de leitura/gravação disponível para o contêiner e é usada para criar arquivos de trabalho, arquivos temporários e arquivos de registro. Esses arquivos são considerados voláteis. Um aplicativo não para de funcionar se eles forem perdidos. A camada de armazenamento de contêiner é exclusiva do contêiner em execução. Por isso, se outro contêiner for criado a partir da mesma imagem de base, recebe outra camada de leitura/gravação. Isso garante que os recursos de cada contêiner sejam isolados de outros contêineres semelhantes.

O armazenamento efêmero de contêineres não é suficiente para aplicativos que precisam manter dados além da vida do contêiner, como os banco de dados. Para oferecer suporte a esses aplicativos, o administrador deve fornecer um contêiner com armazenamento persistente.

**Figura 3.3: Camadas de contêiner**

Os aplicativos conteinerizados não devem tentar usar o armazenamento de contêineres para armazenar dados persistentes, pois eles não podem controlar por quanto tempo o conteúdo será preservado.

Mesmo que fosse possível manter o armazenamento de contêineres por tempo indeterminado, o sistema de arquivos em camadas não funciona bem para cargas de trabalho E/S intensivas e não seria adequado para a maioria dos aplicativos que exigem armazenamento persistente.

Recuperação de armazenamento

O Podman mantém o armazenamento antigo de contêineres interrompidos disponível para ser usado por operações de solução de problemas, como a análise de mensagens de erro em registros de contêineres com falhas.

Se o administrador precisar recuperar um armazenamento de contêineres antigos, o contêiner poderá ser excluído usando `podman rm container_id`. Esse comando também exclui o armazenamento de contêineres. Você pode encontrar os IDs do contêiner interrompido usando o comando `podman ps -a`.

Preparação do diretório do host

O Podman pode montar diretórios de host dentro de um contêiner em execução. O aplicativo conteinerizado vê esses diretórios de host como parte do armazenamento de contêiner, de modo muito semelhante a como os aplicativos comuns veem um volume de rede remota como parte do sistema de arquivos do host. Porém, esse conteúdo dos diretórios de host não é recuperado

depois que o contêiner é interrompido. Por isso, ele pode ser montado em novos contêineres sempre que necessário.

Por exemplo, um contêiner de banco de dados pode usar um diretório de host para armazenar arquivos de banco de dados. Se esse contêiner de banco de dados falhar, o Podman pode criar um novo contêiner usando o mesmo diretório de host e mantendo os dados do banco de dados disponíveis para aplicativos de clientes. Para esse contêiner de banco de dados, não importa onde esse diretório de host está armazenado do ponto de vista do host. Ele poderia estar em qualquer lugar, de uma partição de disco rígido local a um sistema de arquivos de rede remota.

Um contêiner é executado como um processo do sistema operacional do host, com um usuário e um ID de grupo do sistema operacional do host. Por isso, as necessidades do diretório do host precisam ser configuradas permitindo acesso ao contêiner. No RHEL, o diretório do host também precisa ser configurado com o contexto SELinux apropriado, que é `container_file_t`. O Podman usa o contexto do SELinux `container_file_t` para restringir quais arquivos do sistema no host o contêiner tem permissão para acessar. Isso evita o vazamento de informações entre o sistema do host e os aplicativos em execução dentro de contêineres.

Uma forma de configurar o diretório do host é a descrita abaixo:

1. Criar um diretório:

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

2. O usuário que executa os processos no contêiner deve ser capaz de gravar arquivos no diretório. A permissão deverá ser definida com o ID numérico do usuário (UID) a partir do contêiner. Para o serviço MySQL fornecido pela Red Hat, a UID é 27. O comando `podman unshare` fornece uma sessão para executar comandos no mesmo namespace de usuário que o processo em execução no contêiner.

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

3. Aplique o contexto `container_file_t` ao diretório (e todos os subdiretórios) para permitir o acesso de contêineres a todo o seu conteúdo.

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

4. Aplique a política de contêineres do SELinux que você configurou na primeira etapa no diretório criado recentemente:

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

O diretório de host deve ser configurado *antes* de iniciar o contêiner que usa o diretório.

Montagem de um volume

Depois de criado e configurado o diretório do host, a próxima etapa é montar esse diretório em um contêiner. Para montar de bind um diretório de host em um contêiner, adicione a opção `-v` ao comando `podman run`, especificando o caminho do diretório do host e o caminho do armazenamento de contêiner, separados por dois pontos (:).

Por exemplo, para usar o diretório do host `/home/student/dbfiles` para arquivos do banco de dados do servidor MySQL, que são esperados no `/var/lib/mysql` em uma imagem de contêiner do MySQL chamada `mysql`, use o seguinte comando:

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

Nesse comando, se `/var/lib/mysql` já existir na imagem de contêiner `mysql`, a montagem `/home/student/dbfiles` se sobrepõe, mas não remove o conteúdo da imagem de contêiner. Se a montagem for removida, o conteúdo original ficará acessível novamente.

► Exercício Guiado

Criar contêiner MySQL com banco de dados persistente

Neste exercício, você criará um contêiner que armazena o banco de dados do MySQL em um diretório de host.

Resultados

Você deverá ser capaz de implantar um contêiner com um banco de dados persistente.

Antes De Começar

A workstation não deverá ter imagens de contêiner em execução.

Na workstation, execute o seguinte comando:

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

Instruções

- 1. Crie o diretório /home/student/local/mysql com as permissões e o contexto SELinux corretos.
 - 1.1. Crie o diretório /home/student/local/mysql.

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

- 1.2. Adicione o contexto do SELinux apropriado ao diretório /home/student/local/mysql e seu conteúdo.

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

- 1.3. Aplique a política do SELinux ao diretório recém-criado.

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

- 1.4. Verifique se o tipo de contexto do SELinux para o diretório /home/student/local/mysql é container_file_t.

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

15. Altere o proprietário do diretório /home/student/local/mysql para o usuário mysql e o grupo mysql:

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

**nota**

O usuário que executa os processos no contêiner deve ser capaz de gravar arquivos no diretório.

A permissão deverá ser definida com o ID numérico do usuário (UID) a partir do contêiner. Para o serviço MySQL fornecido pela Red Hat, a UID é 27. O comando `podman unshare` fornece uma sessão para executar comandos no mesmo namespace de usuário que o processo em execução no contêiner.

- ▶ 2. Crie uma instância de contêiner MySQL com armazenamento persistente.
 - 2.1. Faça login no Red Hat Container Catalog com sua conta da Red Hat. Se precisar se registrar na Red Hat, consulte as instruções em *Apêndice D, Criação de uma conta da Red Hat*.

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

- 2.2. Faça pull da imagem de contêiner MySQL.

```
[student@workstation ~]$ podman pull registry.redhat.io/rhel8/mysql-80:1  
Trying to pull ...output omitted...registry.redhat.io/rhel8/mysql-80:1...output  
omitted...  
...output omitted...  
Writing manifest to image destination  
Storing signatures  
4ae3a3f4f409a8912cab9fbf71d3564d011ed2e68f926d50f88f2a3a72c809c5
```

- 2.3. Crie um novo contêiner especificando o ponto de montagem para armazenar os dados do banco de dados do MySQL:

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

Esse comando monta o diretório /home/student/local/mysql do host para o diretório /var/lib/mysql/data no contêiner. Por padrão, o banco de dados MySQL armazena os dados no diretório /var/lib/mysql/data.

- 2.4. Verifique se o contêiner foi iniciado corretamente.

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

- 3. Verifique se o diretório /home/student/local/mysql contém um diretório items:

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

O diretório items armazena os dados relacionados ao banco de dados items que foi criado por esse contêiner. Se o diretório items não existir, o ponto de montagem não foi definido corretamente na criação do contêiner.



nota

Como alternativa, você pode executar o mesmo comando com podman unshare para verificar a ID numérica do usuário (UID) do contêiner.

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

Encerramento

Na workstation, execute o script lab manage-storage finish para concluir esse laboratório.

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

Isso conclui o exercício.

Acesso aos contêineres

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Descrever o básico das redes com contêineres.
- Conectar-se remotamente a serviços no contêiner.

Mapeamento de portas de redes

Acessar um contêiner sem raiz da rede do host pode ser desafiador porque não há endereços IP disponíveis para um contêiner sem raiz.

Para resolver esses problemas, defina as regras de encaminhamento de porta para permitir acesso externo a um serviço de contêiner. Use a opção `-p [<IP address>:]<host port>:<container port>` com o comando `podman run` para criar um contêiner acessível externamente.

Considere o exemplo a seguir:

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

Este exemplo cria um contêiner acessível externamente. O valor que 8080:80 especifica que todas as solicitações para a porta 8080 no host são encaminhadas para a porta 8080 no contêiner

Você também pode usar a opção `-p` para vincular a porta ao endereço IP especificado.

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

Este exemplo limita o acesso externo ao contêiner apache2 para solicitações do `localhost` para a porta 8081 do host. Essas solicitações são encaminhadas para a porta 8080 no contêiner apache2.

Se uma porta não for especificada para a porta do host, o Podman atribui uma porta de host disponível aleatoriamente:

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

Para ver a porta atribuída pelo Podman, use o comando `podman port <container name>`:

```
[user@host ~]$ podman port apache3
8080/tcp -> 127.0.0.1:35134
[user@host ~]$ curl 127.0.0.1:35134
```

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

Se apenas uma porta do contêiner for especificada com a opção `-p`, uma porta do host disponível é atribuída ao contêiner aleatoriamente. As solicitações para essa porta do host atribuída de qualquer endereço IP são encaminhadas para a porta do contêiner.

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

Neste exemplo, qualquer solicitação roteável para a porta 37068 do host é encaminhada para a porta 8080 no contêiner.



Referências

Container Network Interface - networking for Linux containers

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

Cloud Native Computing Foundation

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

► Exercício Guiado

Carregamento do banco de dados

Neste exercício, você criará um contêiner de banco de dados MySQL com encaminhamento de portas ativado. Depois de preencher um banco de dados com um script SQL, você verificará o conteúdo do banco de dados usando três métodos diferentes.

Resultados

Você deverá ser capaz de implantar um contêiner de banco de dados e carregar um script SQL.

Antes De Começar

Abra um terminal na máquina `workstation` como usuário `student` e execute o seguinte comando:

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

Isso garante que o diretório `/home/student/local/mysql` existe e que está configurado com as permissões corretas para ativar o armazenamento persistente para o contêiner do MySQL.

Instruções

- 1. Crie uma instância de contêiner MySQL com armazenamento persistente e encaminhamento de portas.
 - 1.1. Faça login no Red Hat Container Catalog com sua conta da Red Hat. Se precisar se registrar na Red Hat, consulte as instruções em *Apêndice D, Criação de uma conta da Red Hat*.

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

- 1.2. Faça download da imagem de contêiner de banco de dados do MySQL e, depois, crie uma instância de contêiner MySQL com armazenamento persistente e encaminhamento de portas.

```
[student@workstation ~]$ podman run --name mysqldb-port \
> -d -v /home/student/local/mysql:/var/lib/mysql/data -p 13306:3306 \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
Trying to pull ...output omitted...
Copying blob sha256:1c9f515...output omitted...
 72.70 MB / ? [=====] 5s
Copying blob sha256:1d2c4ce...output omitted...
```

```
1.54 KB / ? [-----] 0s
Copying blob sha256:f1e961f...output omitted...
6.85 MB / ? [-----] 0s
Copying blob sha256:9f1840c...output omitted...
62.31 MB / ? [-----] 7s
Copying config sha256:60726...output omitted...
6.85 KB / 6.85 KB [=====] 0s
Writing manifest to image destination
Storing signatures
066630d45cb902ab533d503c83b834aa6a9f9cf88755cb68eedb8a3e8edbc5aa
```

A última linha da sua saída e o tempo necessário para fazer download de cada camada de imagem serão diferentes.

A opção `-p` configura o encaminhamento de portas para que a porta 13306 no host local seja encaminhada para o contêiner da porta 3306.



nota

O script inicial cria o diretório `/home/student/local/mysql` com a propriedade adequada e o contexto SELinux exigido pelo banco de dados conteinerizado.

- ▶ 2. Verifique se o contêiner `mysql ldb-port` foi iniciado com êxito e ativa o encaminhamento de portas.

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

- ▶ 3. Preencha o banco de dados usando o arquivo fornecido. Se não houver erros, o comando não retornará nenhuma saída.

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

Há várias maneiras de verificar se o banco de dados foi carregado com êxito. As próximas etapas mostram três métodos diferentes. Você só precisa usar um dos métodos.

- ▶ 4. Verifique se o banco de dados foi carregado com êxito executando um comando não interativo dentro do contêiner.

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

- 5. Verifique se o banco de dados foi carregado com êxito usando o encaminhamento de portas do host local. Esse método alternativo é opcional.

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

- 6. Verifique se o banco de dados foi carregado com êxito executando uma sessão de terminal interativa dentro do contêiner. Esse método alternativo é opcional.

- 6.1. Abra um shell Bash dentro do contêiner.

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

- 6.2. Verifique se o banco de dados contém dados:

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

- 6.3. Saia do contêiner:

```
bash-4.4$ exit
```

Encerramento

Na workstation, execute o script lab manage-networking finish para concluir esse laboratório.

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

Isso conclui o exercício.

► Laboratório Aberto

Gerenciamento de contêineres

Resultados

Você deverá ser capaz de implantar e gerenciar um banco de dados persistente usando um volume compartilhado. Você também deverá ser capaz de iniciar um segundo banco de dados usando o mesmo volume compartilhado e observar que os dados são consistentes entre os dois contêineres, pois estão usando o mesmo diretório no host para armazenar os dados MySQL.

Antes De Começar

Abra um terminal na `workstation` como usuário `student` e execute o seguinte comando:

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

Instruções

1. Crie o diretório `/home/student/local/mysql` com as permissões e o contexto SELinux corretos.
2. Implante uma instância de contêiner MySQL usando as seguintes características:
 - *Nome:* mysql-1
 - *Executar como daemon:* sim
 - *Volume:* da pasta do host `/home/student/local/mysql` para a pasta do contêiner `/var/lib/mysql/data`
 - *Imagem do contêiner:* `registry.redhat.io/rhel8/mysql-80:1`
 - *Encaminhamento de porta:* sim, da porta de host 13306 para a porta de contêiner 3306
 - *Variáveis de ambiente:*
 - `MYSQL_USER:user1`
 - `MYSQL_PASSWORD:mypa55`
 - `MYSQL_DATABASE:items`
 - `MYSQL_ROOT_PASSWORD:r00tpa55`
3. Carregue o banco de dados `items` usando o script `/home/student/D0180/labs/manage-review/db.sql`.
4. Interrompa o contêiner normalmente.

**Importante**

Essa etapa é muito importante, pois um novo contêiner será criado compartilhando o mesmo volume dos dados do banco de dados. Ter dois contêineres usando o mesmo volume pode corromper o banco de dados. Não reinicie o contêiner `mysql-1`.

5. Crie um novo contêiner com as seguintes características:
 - *Nome:* `mysql-2`
 - *Executar como um daemon:* sim
 - *Volume:* da pasta do host `/home/student/local/mysql` para a pasta do contêiner `/var/lib/mysql/data`
 - *Imagen do contêiner:* `registry.redhat.io/rhel8/mysql-80:1`
 - *Encaminhamento de porta:* sim, da porta de host 13306 para a porta de contêiner 3306
 - *Variáveis de ambiente:*
 - `MYSQL_USER:user1`
 - `MYSQL_PASSWORD:mypa55`
 - `MYSQL_DATABASE:items`
 - `MYSQL_ROOT_PASSWORD:r00tpa55`
6. Salve a lista de todos os contêineres (incluindo os interrompidos) no arquivo `/tmp/my-containers`.
7. Acesse o shell Bash no contêiner e verifique se o banco de dados `items` e a tabela `Item` ainda estão disponíveis. Confirme também que a tabela contém dados.
8. Usando encaminhamento de portas, insira uma nova linha na tabela `Item`. A linha deve ter um valor `description` de `Finished lab` e um valor `done` de 1.
9. Como o primeiro contêiner não é mais necessário, remova-o para liberar recursos.

Avaliação

Avalie seu trabalho executando o comando `lab manage-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `workstation`, execute o comando `lab manage-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

► Solução

Gerenciamento de contêineres

Resultados

Você deverá ser capaz de implantar e gerenciar um banco de dados persistente usando um volume compartilhado. Você também deverá ser capaz de iniciar um segundo banco de dados usando o mesmo volume compartilhado e observar que os dados são consistentes entre os dois contêineres, pois estão usando o mesmo diretório no host para armazenar os dados MySQL.

Antes De Começar

Abra um terminal na `workstation` como usuário `student` e execute o seguinte comando:

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

Instruções

- Crie o diretório `/home/student/local/mysql` com as permissões e o contexto SELinux corretos.

1. Crie o diretório `/home/student/local/mysql`.

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

2. Adicione o contexto do SELinux apropriado ao diretório `/home/student/local/mysql` e seu conteúdo. Com o contexto correto, é possível montar esse diretório em um contêiner em execução.

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

3. Aplique a política do SELinux ao diretório recém-criado.

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

4. Altere o proprietário do diretório `/home/student/local/mysql` para corresponder ao usuário `mysql` e ao grupo `mysql` da imagem de contêiner `registry.redhat.io/rhel8/mysql-80:1`.

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

2. Implante uma instância de contêiner MySQL usando as seguintes características:

- *Nome:* `mysql-1`

- Executar como daemon: sim
- Volume: da pasta do host /home/student/local/mysql para a pasta do contêiner /var/lib/mysql/data
- Imagem do contêiner: registry.redhat.io/rhel8/mysql-80:1
- Encaminhamento de porta: sim, da porta de host 13306 para a porta de contêiner 3306
- Variáveis de ambiente:
 - MYSQL_USER:user1
 - MYSQL_PASSWORD:mypa55
 - MYSQL_DATABASE:items
 - MYSQL_ROOT_PASSWORD:r00tpa55

2.1. Faça login no Red Hat Container Catalog com sua conta da Red Hat.

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

2.2. Crie e inicie o contêiner.

```
[student@workstation ~]$ podman run --name mysql-1 -p 13306:3306 \
> -d -v /home/student/local/mysql:/var/lib/mysql/data \
> -e MYSQL_USER=user1 -e MYSQL_PASSWORD=mypa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
Trying to pull ...output omitted...
...output omitted...
Writing manifest to image destination
Storing signatures
616azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cd
```

2.3. Verifique se o contêiner foi iniciado corretamente.

```
[student@workstation ~]$ podman ps --format="{{.ID}} {{.Names}}"
616azfaa55x8    mysql-1
```

3. Carregue o banco de dados items usando o script /home/student/D0180/labs/manage-review/db.sql.
 - 3.1. Carregue o banco de dados usando os comandos SQL em /home/student/D0180/labs/manage-review/db.sql.

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

- 3.2. Use uma instrução SELECT do SQL para produzir todas as linhas da tabela Item a fim de verificar se o banco de dados de Items está carregado.

**nota**

Você pode adicionar o parâmetro -e SQL ao comando mysql e executar uma instrução SQL.

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

4. Interrompa o contêiner normalmente.

**Importante**

Essa etapa é muito importante, pois um novo contêiner será criado compartilhando o mesmo volume dos dados do banco de dados. Ter dois contêineres usando o mesmo volume pode corromper o banco de dados. Não reinicie o contêiner mysql-1.

- 4.1. Use o comando podman para parar o contêiner.

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

5. Crie um novo contêiner com as seguintes características:

- *Nome:* mysql-2
- *Executar como um daemon:* sim
- *Volume:* da pasta do host /home/student/local/mysql para a pasta do contêiner /var/lib/mysql/data
- *Imagem do contêiner:* registry.redhat.io/rhel8/mysql-80:1
- *Encaminhamento de porta:* sim, da porta de host 13306 para a porta de contêiner 3306
- *Variáveis de ambiente:*
 - MYSQL_USER:user1
 - MYSQL_PASSWORD:mypa55
 - MYSQL_DATABASE:items
 - MYSQL_ROOT_PASSWORD:r00tpa55

- 5.1. Crie e inicie o contêiner.

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

- 5.2. Verifique se o contêiner foi iniciado corretamente.

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

6. Salve a lista de todos os contêineres (incluindo os interrompidos) no arquivo /tmp/my-containers.

- 6.1. Use o comando podman para salvar as informações em /tmp/my-containers.

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

7. Acesse o shell Bash no contêiner e verifique se o banco de dados `items` e a tabela `Item` ainda estão disponíveis. Confirme também que a tabela contém dados.

- 7.1. Acesse o shell Bash dentro do contêiner.

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

- 7.2. Conecte-se ao servidor MySQL.

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

- 7.3. Liste todos os bancos de dados e confirme que o banco de dados `items` está disponível.

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

- 7.4. Liste todas as tabelas do banco de dados `items` e verifique se a tabela `Item` está disponível.

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

7.5. Exiba os dados da tabela.

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

7.6. Saia do cliente MySQL e do shell do contêiner.

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

8. Usando encaminhamento de portas, insira uma nova linha na tabela `Item`. A linha deve ter um valor `description` de `Finished lab` e um valor `done` de `1`.

8.1. Conecte-se ao banco de dados MySQL.

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

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

mysql>
```

8.2. Insira a nova linha.

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

8.3. Saia do cliente MySQL.

```
mysql> exit
Bye
```

9. Como o primeiro contêiner não é mais necessário, remova-o para liberar recursos.

9.1. Use o seguinte comando para remover o contêiner:

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

Avaliação

Avalie seu trabalho executando o comando `lab manage-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `workstation`, execute o comando `lab manage-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

Sumário

Neste capítulo, você aprendeu que:

- O Podman tem subcomandos para: criar um novo contêiner (`run`), excluir um contêiner (`rm`), listar contêineres (`ps`), interromper um contêiner (`stop`) e iniciar um processo em um contêiner (`exec`).
- O armazenamento padrão de contêineres é efêmero, o que significa que seu conteúdo não estará presente depois que o contêiner é removido.
- Os contêineres podem usar uma pasta a partir do sistema de arquivos do host para trabalhar com dados persistentes.
- O Podman monta volumes em um contêiner com a opção `-v` no comando `podman run`.
- O comando `podman exec` inicia um processo adicional dentro de um contêiner em execução:
- O Podman mapeia as portas locais para portas de contêiner usando a opção `-p` no subcomando `run`.

capítulo 4

Gerenciamento de imagens de contêiner

Meta

Gerenciar o ciclo de vida de uma imagem de contêiner da criação à exclusão.

Objetivos

- Pesquisar e fazer pull de imagens de registros remotos.
- Exportar, importar e gerenciar imagens de contêiner localmente e em um registro.

Seções

- Acesso a registros (e teste)
- Manipulação de imagens de contêiner (e exercício orientado)

Laboratório

- Gerenciamento de imagens de contêiner

Acesso aos registros

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Pesquise e capte imagens de registros remotos usando os comandos do Podman e a API REST do registro.
- Listar as vantagens de usar um registro público certificado para fazer download de imagens seguras.
- Personalizar a configuração do Podman para acessar registros alternativos de imagem de contêiner.
- Listar imagens baixadas de um registro para o sistema de arquivos local.
- Gerenciar tags para extrair imagens marcadas.

Registros públicos

Os registros de imagem são serviços que oferecem imagens de contêiner para download. Eles permitem que criadores de imagens e mantenedores armazenem e distribuam imagens de contêiner para públicos-alvo privados ou públicos.

O Podman pesquisa e faz download de imagens de contêineres de registros públicos e privados. O Red Hat Container Catalog é o registro de imagens públicas gerenciado pela Red Hat. Ele hospeda um enorme conjunto de imagens de contêiner, incluindo aquelas fornecidas por grandes projetos open source, como o Apache, MySQL e Jenkins. Todas as imagens no Container Catalog são examinadas pela equipe de segurança interna da Red Hat, e todos os componentes foram reconstruídos pela Red Hat para evitar vulnerabilidades de segurança conhecidas.

As imagens de contêiner da Red Hat oferecem os seguintes benefícios:

- *Fonte confiável*: a Red Hat conhece e confia em todas as imagens de contêiner.
- *Dependências originais*: nenhum dos pacotes de contêiner foi manipulado, e eles incluem somente bibliotecas conhecidas.
- *Sem vulnerabilidades*: as imagens de contêiner são livres de vulnerabilidades conhecidas nos componentes ou camadas da plataforma.
- *Proteção em tempo de execução* : Todos os aplicativos em imagens de contêiner são executados como non-root usuários, minimizando a superfície de exposição a aplicativos maliciosos ou defeituosos.
- Compatível com o *Red Hat Enterprise Linux (RHEL)*: as imagens de contêiner são compatíveis com todas as plataformas do RHEL, de bare-metal à nuvem.
- *Suporte da Red Hat*: a Red Hat dá suporte comercial à toda a pilha.

Quay.io é outro repositório de imagens públicas patrocinado pela Red Hat. O Quay.io introduz vários recursos interessantes, como criação de imagem no lado do servidor, controles de acesso refinados e verificação automática de imagens para vulnerabilidades conhecidas.

Embora as imagens do Red Hat Container Catalog sejam confiáveis e verificadas, o Quay.io oferece imagens dinâmicas regularmente atualizadas por criadores. Os usuários do Quay.io podem criar seus namespaces, com um controle de acesso refinado, e publicar as imagens que eles criam para esse namespace. Os usuários do Container Catalog raramente ou nunca enviam novas imagens, mas consomem imagens confiáveis geradas pela equipe da Red Hat.

Registros privados

Criadores ou mantenedores de imagens desejam disponibilizar suas imagens publicamente. Porém, há outros criadores de imagens que preferem manter suas imagens privadas devido a:

- Privacidade da empresa e proteção de segredos.
- Leis e restrições legais.
- Evitar a publicação de imagens em desenvolvimento.

Em alguns casos, as imagens privadas são preferenciais. Registros privados dão aos criadores de imagem o controle sobre o posicionamento, a distribuição e o uso de suas imagens.

Configuração de registros no Podman

Para configurar registros para o comando `podman`, é necessário atualizar o arquivo `/etc/containers/registries.conf`. Edite a entrada `registries` na seção `[registries.search]`, adicionando uma entrada à lista de valores:

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



nota

Use um FQDN e um número de porta para identificar um registro. Um registro que não inclui um número de porta tem um número de porta padrão 5000. Se o registro usar uma porta diferente, ele deverá ser especificado. Indique os números de porta anexando dois-pontos (:) e o número da porta após o FQDN.

Conexões seguras para um registro exigem um certificado confiável. Para suportar conexões inseguras, adicione o nome do registro à entrada `registries` na seção `[registries.insecure]` do arquivo `/etc/containers/registries.conf`:

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

Acesso aos registros

O Podman fornece comandos que permitem que você pesquise imagens. Registros de imagens também podem ser acessados por uma API. A seguir discutimos as duas abordagens.

Pesquisa de imagens em registros

O comando `podman search` encontra imagens por nome de imagem, nome de usuário ou descrição a partir de todos os registros listados no arquivo de configuração `/etc/containers/registries.conf`. A sintaxe básica para o comando `podman search` é mostrada abaixo:

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

A seguinte tabela mostra opções úteis disponíveis para o subcomando search:

Opção	Descrição
--limit <number>	Limita o número de imagens listadas por registry.
--filter <filter=value>	Saída de filtro com base nas condições fornecidas. Os filtros compatíveis são: stars=<number> : exibir apenas imagens com pelo menos esse número de estrelas. is-automated=<true false> : exibir apenas imagens criadas automaticamente. is-official=<true false> : exibir apenas imagens marcadas como oficiais.
--tls-verify <true false>	Ativa ou desativa a validação do certificado HTTPS para todos os registries usados.
--list-tags	Listar as tags disponíveis no repositório para a imagem especificada.

API de registro HTTP

Um registro remoto expõe serviços da web que fornecem uma interface de programação de aplicativos (API) ao registro. O Podman usa essas interfaces para acessar e interagir com repositórios remotos. Muitos registros estão em conformidade com a especificação Docker Registry HTTP API v2, que expõe uma interface REST padronizada para interação com o registro. Você pode usar essa interface REST para interagir diretamente com um registro, em vez de usar o Podman.

Exemplos de uso dessa API com os comandos curl são mostrados abaixo:

Para listar todos os repositórios disponíveis em um registro, use o ponto de extremidade /v2/_catalog. O parâmetro n é usado para limitar o número de repositórios a serem retornados:

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

**nota**

Se o Python estiver disponível, use-o para formatar a resposta JSON:

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

O ponto de extremidade /v2/<name>/tags/list fornece a lista de tags disponíveis para uma única imagem:

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

**nota**

O Quay.io oferece uma API dedicada para interagir com os repositórios além do que é especificado na API do repositório Docker. Consulte <https://docs.quay.io/api/> para obter detalhes.

Autenticação do registro

Alguns registros de imagem de contêiner exigem autorização de acesso. O comando podman login permite autenticação de nome de usuário e senha para um registro:

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

A API de registro HTTP exige credenciais de autenticação. Primeiro, use o serviço Red Hat Single Sign On (SSO) para obter um token de acesso:

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

Depois, inclua esse token em um cabeçalho de autorização **Bearer** em solicitações subsequentes:

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



nota

Outros registros podem exigir etapas diferentes para fornecer credenciais. Se um registro aderir à autenticação Docker Registry HTTP v2 API em conformidade com o esquema RFC7235.

Pull de imagens

Para extrair imagens de contêiner de um registro, use o comando `podman pull`:

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

O comando `podman pull` usa o nome da imagem obtido do subcomando `search` para extrair uma imagem de um registro. O subcomando `pull` permite adicionar o nome de registro à imagem. Essa variante é compatível com ter a mesma imagem em vários registros.

Por exemplo, para extrair um contêiner NGINX do registro `quay.io`, use o seguinte comando:

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



nota

Se o nome da imagem não incluir um nome de registro, o Podman procurará uma imagem de contêiner correspondente usando os registros listados no arquivo de configuração `/etc/containers/registries.conf`. O Podman procura por imagens nos registros na mesma ordem em que eles aparecem no arquivo de configuração.

Listagem de cópias de imagens locais

Qualquer imagem de contêiner baixada de um registro é armazenada localmente no mesmo host em que comando podman é executado. Esse comportamento evita a repetição de downloads de imagens e minimiza o tempo de implantação de um contêiner. O Podman também armazena todas as imagens de contêiner personalizado que você cria no mesmo armazenamento local.



nota

Por padrão, o Podman armazena imagens de contêiner no diretório `/var/lib/containers/storage/overlay-images`.

O Podman fornece um subcomando `images` para listar todas as imagens do contêiner armazenadas localmente.

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

Tags de imagem

Uma tag de imagem é um mecanismo para dar suporte a várias versões de uma imagem. Este recurso é usado quando múltiplas versões do mesmo software são fornecidas, como um contêiner pronto para produção ou as atualizações mais recentes do mesmo software desenvolvido para avaliação da comunidade. Qualquer subcomando do Podman que requer um nome de imagem de contêiner aceita um parâmetro de tag para diferenciar entre várias tags. Se um nome de imagem não contiver uma tag, o valor da tag será padronizado como `latest`. Por exemplo, para extrair uma imagem com a tag `1` de `rhel8/mysql-80`, use o seguinte comando:

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

Para iniciar um novo contêiner baseado na imagem `rhel8/mysql-80:1`, use o seguinte comando:

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



Referências

Red Hat Container Catalog

<https://registry.redhat.io>

Quay.io

<https://quay.io>

Docker Registry HTTP API V2

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

RFC7235 - HTTP/1.1: Autenticação

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

► Teste

Trabalho com registros

Escolha as respostas corretas para as perguntas a seguir, com base nas seguintes informações:

O Podman está disponível em um host RHEL com a seguinte entrada no arquivo `/etc/containers/registries.conf`:

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

Os hosts `registry.redhat.io` e `quay.io` têm um registro em execução, ambos possuem certificados válidos e usam a versão 1 do registro. As imagens a seguir estão disponíveis para cada host:

Nomes/tags de imagem por registro

Registro	Imagen
registry.redhat.io	nginx / 1.1.6
	mysql / 8.0
	httpd/2.4
quay.io	mysql / 5.7
	httpd/2.4

Nenhuma imagem está disponível localmente.

- ▶ 1. Quais comandos exibem imagens mysql disponíveis para download em `registry.redhat.io`? (Escolha duas opções.)
 - a. podman search registry.redhat.io/mysql
 - b. podman images
 - c. podman pull mysql
 - d. podman search mysql

- ▶ 2. Qual comando é usado para listar todas as tags de imagem disponíveis para a imagem de contêiner httpd?
 - a. podman search --list-tags httpd
 - b. podman images httpd
 - c. podman pull --all-tags=true httpd
 - d. There is no podman command available to search for tags.

► 3. Quais os dois comandos que extraem a imagem httpd com o marcador 2.4? (Escolha duas opções.)

- a. podman pull httpd:2.4
- b. podman pull httpd:latest
- c. podman pull quay.io/httpd
- d. podman pull registry.redhat.io/httpd:2.4

► 4. Ao executar os comandos a seguir, quais imagens de contêiner serão baixadas?

```
podman pull registry.redhat.io/httpd:2.4  
podman pull quay.io/mysql:8.0
```

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

► Solução

Trabalho com registros

Escolha as respostas corretas para as perguntas a seguir, com base nas seguintes informações:

O Podman está disponível em um host RHEL com a seguinte entrada no arquivo `/etc/containers/registries.conf`:

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

Os hosts `registry.redhat.io` e `quay.io` têm um registro em execução, ambos possuem certificados válidos e usam a versão 1 do registro. As imagens a seguir estão disponíveis para cada host:

Nomes/tags de imagem por registro

Registro	Imagen
registry.redhat.io	nginx / 1.1.6
	mysql / 8.0
	httpd/2.4
quay.io	mysql / 5.7
	httpd/2.4

Nenhuma imagem está disponível localmente.

- ▶ 1. Quais comandos exibem imagens mysql disponíveis para download em `registry.redhat.io`? (Escolha duas opções.)
 - a. podman search registry.redhat.io/mysql
 - b. podman images
 - c. podman pull mysql
 - d. podman search mysql

- ▶ 2. Qual comando é usado para listar todas as tags de imagem disponíveis para a imagem de contêiner httpd?
 - a. podman search --list-tags httpd
 - b. podman images httpd
 - c. podman pull --all-tags=true httpd
 - d. There is no podman command available to search for tags.

► 3. Quais os dois comandos que extraem a imagem httpd com o marcador 2.4? (Escolha duas opções.)

- a. podman pull httpd:2.4
- b. podman pull httpd:latest
- c. podman pull quay.io/httpd
- d. podman pull registry.redhat.io/httpd:2.4

► 4. Ao executar os comandos a seguir, quais imagens de contêiner serão baixadas?

```
podman pull registry.redhat.io/httpd:2.4  
podman pull quay.io/mysql:8.0
```

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

Manipulação de imagens de contêiner

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Salvar e carregar imagens de contêiner em arquivos locais.
- Excluir imagens do armazenamento local.
- Criar novas imagens de contêiner a partir de contêineres e atualizar os metadados da imagem.
- Gerenciar as tags de imagens para distribuição.

Introdução

Há várias maneiras de gerenciar contêineres de imagem mantendo a conformidade com os princípios de DevOps. Por exemplo, um desenvolvedor termina de testar um contêiner personalizado na sua máquina e precisa transferir essa imagem de contêiner para outro host para outro desenvolvedor ou para um servidor de produção. Há duas maneiras de fazer isso:

1. Salve a imagem de contêiner em um arquivo `.tar`.
2. Publique (envie) a imagem de contêiner em um registro de imagem.



nota

Uma das maneiras à disposição de um desenvolvedor para criar esse contêiner personalizado é discutida posteriormente neste capítulo (`podman commit`). No entanto, nos capítulos seguintes, discutiremos a maneira recomendada de fazer isso usando `Containerfiles`.

Salvamento e carregamento de imagens

É possível salvar imagens existentes do armazenamento local do Podman em um arquivo `.tar` usando o comando `podman save`. O arquivo gerado não é um arquivo TAR comum: ele contém metadados de imagem e preserva as camadas da imagem original. Usando esse arquivo, o Podman pode recriar a imagem original.

A sintaxe geral do subcomando `save` é a seguinte:

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

O Podman envia a imagem gerada à saída padrão como dados binários. Para evitar isso, use a opção `-o`.

O exemplo a seguir salva a imagem de contêiner MySQL baixada anteriormente do Red Hat Container Catalog para o arquivo `mysql.tar`:

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

Observe o uso da opção `-o` neste exemplo.

Use os arquivos `.tar` gerados pelo subcomando `save` para fins de backup. Para restaurar a imagem de contêiner, use o comando `podman load`. A sintaxe geral do comando é a seguinte:

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

O comando `load` é o oposto do comando `save`.

Por exemplo, esse comando carregaria uma imagem salva em um arquivo chamado `mysql.tar`.

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

Se o arquivo `.tar` dado como argumento não for uma imagem de contêiner com metadados, ocorrerá falha no comando `podman load`.



nota

Para economizar espaço em disco, compacte o arquivo gerado pelo subcomando `save` com o Gzip usando o parâmetro `--compress`. O subcomando `load` usa o comando `gunzip` antes de importar o arquivo para o armazenamento local.

Exclusão de imagens

O Podman mantém qualquer imagem baixada em seu armazenamento local, mesmo aquelas que atualmente não são utilizadas por nenhum contêiner. Porém, as imagens podem ficar desatualizadas e, assim, precisarão ser substituídas.



nota

As atualizações nas imagens em um registro não são atualizadas automaticamente. A imagem deve ser removida e, então, capturada novamente para garantir que o armazenamento local tenha a versão mais recente de uma imagem.

Para excluir uma imagem do armazenamento local, execute o comando `podman rmi`.

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

Uma imagem pode ser referenciada usando o nome ou a ID para remoção. O Podman não pode excluir imagens enquanto os contêineres estiverem usando essa imagem. Você deve interromper e remover todos os contêineres usando essa imagem antes de excluí-la.

Para evitar isso, o subcomando `rmi` tem a opção `--force`. Essa opção força a remoção de uma imagem mesmo que ela seja usada por vários contêineres ou que esses contêineres estejam em execução. O Podman interrompe e remove todos os contêineres usando a imagem removida forçadamente antes de removê-la.

Exclusão de todas as imagens

Para excluir todas as imagens que não são usadas por nenhum contêiner, use o seguinte comando:

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

O comando retorna todas as IDs de imagem disponíveis no armazenamento local e as envia como parâmetros do comando `podman rmi` para remoção. As imagens que estão em uso não são excluídas. Porém, isso não impede que imagens não usadas sejam removidas.

Modificação de imagens

Idealmente, todas as imagens de contêiner devem ser compiladas usando um `Containerfile` para criar um conjunto de camadas de imagem limpo e leve sem arquivos de log, arquivos temporários ou outros artefatos criados pela personalização de contêineres. No entanto, alguns usuários podem fornecer imagens de contêiner como elas estão, sem um `Containerfile`. Como abordagem alternativa para a criação de novas imagens, altere um contêiner em execução e salve suas camadas para criar uma nova imagem de contêiner. O comando `podman commit` fornece esse recurso.



Cuidado

Embora o comando `podman commit` seja a abordagem mais simples para criar novas imagens, ele não é recomendado em função do tamanho da imagem (`commit` mantém os logs e arquivos de ID de processo nas camadas capturadas) e da falta de rastreamento de alterações. Um `Containerfile` oferece um mecanismo robusto para personalizar e implementar alterações em um contêiner usando um conjunto legível de comandos sem o conjunto de arquivos que são gerados pelo sistema operacional.

A sintaxe para o comando `podman commit` é a seguinte:

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

A tabela a seguir mostra as opções mais importantes disponíveis para o comando `podman commit`:

Opção	Descrição
<code>--author ""</code>	Identifica quem cria a imagem de contêiner.
<code>--message ""</code>	Inclui uma mensagem de commit no registro.
<code>--format</code>	Seleciona o formato da imagem. As opções válidas são <code>oci</code> e <code>docker</code> .



nota

A opção `--message` não está disponível no formato de contêiner OCI padrão.

Para encontrar a ID de um contêiner em execução no Podman, execute o comando `podman ps`:

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

No fim, os administradores poderão personalizar a imagem e definir o contêiner no estado desejado. Para identificar quais arquivos foram alterados, criados ou excluídos desde que o contêiner foi iniciado, use o subcomando `diff`. Esse subcomando exige apenas o nome e a ID do contêiner:

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

O subcomando `diff` marca os arquivos adicionados com um A, os arquivos alterados com um C e os arquivos excluídos com um D.



nota

O comando `diff` apenas relata os arquivos adicionados, alterados ou excluídos ao sistema de arquivos de contêiner. Os arquivos montados em um contêiner em execução não são considerados como parte do sistema de arquivos de contêiner. Para recuperar a lista de arquivos e diretórios montados de um contêiner em execução, use o comando `podman inspect`:

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

Os arquivos nessa lista e os arquivos localizados em diretórios nessa lista não serão exibidos na saída do comando `podman diff`.

Para salvar as alterações em outra imagem, execute o seguinte comando:

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

O exemplo anterior cria uma nova imagem chamada `mysql-custom`.

Marcação de imagens

Um projeto com várias imagens baseado no mesmo software pode ser distribuído, criando projetos individuais para cada imagem. Porém, essa abordagem exige mais manutenção para gerenciar e implantar as imagens nos locais corretos.

Os registros de imagens de contêiner são compatíveis com tags para distinguir várias versões do mesmo projeto. Por exemplo, um cliente pode usar uma imagem de contêiner executada com um banco de dados MySQL ou PostgreSQL e usar uma tag como forma de diferenciar quais bancos de dados são usados por uma imagem de contêiner.

**nota**

Normalmente, as tags são usadas pelos desenvolvedores de contêineres para diferenciar entre várias versões do mesmo software. Várias tags são fornecidas para identificar uma versão facilmente. O site oficial da imagem de contêiner MySQL usa a versão como o nome de tag (8.0). Além disso, a mesma imagem pode ter uma segunda tag com a versão menor para minimizar a necessidade de obter uma versão específica.

Para marcar uma imagem com tag, use o comando `podman tag`:

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

O argumento `IMAGE` é o nome da imagem com uma tag opcional que é gerenciada pelo Podman. O argumento a seguir se refere ao novo nome alternativo para a imagem. O Podman presume que se trata da versão mais recente, conforme indicado pela tag `latest`, se o valor da tag estiver ausente. Por exemplo, para marcar uma imagem, use o seguinte comando:

```
[user@host ~]$ podman tag mysql-custom devops/mysql
```

A opção `mysql-custom` corresponde ao nome da imagem no registro do contêiner.

Para usar um nome de tag diferente, use o seguinte comando:

```
[user@host ~]$ podman tag mysql-custom devops/mysql:snapshot
```

Remoção de tags de imagens

Uma única imagem pode ter várias tags atribuídos usando o comando `podman tag`. Para removê-las, use o comando `podman rmi`, conforme mencionado anteriormente:

```
[user@host ~]$ podman rmi devops/mysql:snapshot
```

**nota**

Como várias tags podem apontar para a mesma imagem, para remover uma imagem referenciada por várias tags, remova cada tag deve individualmente primeiro.

Práticas recomendadas para marcar imagens

O Podman adiciona automaticamente a tag `latest` se você não especificar nenhuma tag, pois o Podman considera que a imagem seja o build mais recente. Entretanto, isso pode não acontecer, dependendo de como cada projeto usa as tags. Por exemplo, muitos projetos de open source consideram que a tag `latest` corresponde à versão mais recente, mas não à compilação mais recente.

Além disso, várias tags são fornecidas para minimizar a necessidade de lembrar a versão mais recente de uma versão específica de um projeto. Assim, se houver uma versão de projeto (por

exemplo, 2.1.10), outra tag chamada 2.1 poderá ser criada apontando para a mesma imagem da versão 2.1.10. Isso simplifica a captura de imagens do registro.

Publicação de imagens em um registro

Para publicar uma imagem em um registro, ela deve estar no armazenamento local do Podman e deve estar marcada com tag para identificação. Para enviar a imagem ao registro, a sintaxe do subcomando push é:

```
[user@host ~]$ podman push [OPTIONS] IMAGE [DESTINATION]
```

Por exemplo, para enviar a imagem bitnami/nginx ao seu repositório, use o seguinte comando:

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

O exemplo anterior envia a imagem para Quay.io.



Referências

[Site do Podman](#)

<https://podman.io/>

► Exercício Guiado

Criação de uma imagem personalizada de contêiner do Apache

Neste exercício orientado, você criará uma imagem personalizada de contêiner do Apache usando o comando `podman commit`.

Resultados

Você deverá ser capaz de criar uma imagem personalizada de contêiner.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Abra um terminal na máquina `workstation` como usuário `student` e execute o seguinte comando:

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

Instruções

- 1. Faça login na sua conta do Quay.io e inicie um contêiner usando a imagem disponível em `quay.io/redhat/training/httpd-parent`. A opção `-p` permite que você especifique uma porta de redirecionamento. Nesse caso, o Podman encaminha as solicitações recebidas na porta TCP 8180 do host para a porta TCP 80 do contêiner.

```
[student@workstation ~]$ podman login quay.io
Username: your_quay_username
Password: your_quay_password
Login Succeeded!
[student@workstation ~]$ podman run -d --name official-httpd \
> -p 8180:80 quay.io/redhat/training/httpd-parent
...output omitted...
Writing manifest to image destination
Storing signatures
`3a6baecaff2b`4e8c53b026e04847dda5976b773ade1a3a712b1431d60ac5915d
```

A última linha da saída é diferente da última linha mostrada acima. Observe os primeiros doze caracteres.

- 2. Crie uma página HTML no contêiner `official-httpd`.
 - 2.1. Acesse o shell do contêiner usando o subcomando `podman exec` e crie uma página HTML.

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

2.2. Saia do contêiner.

```
bash-4.4# exit  
exit
```

2.3. Certifique-se de que o arquivo HTML seja acessível a partir da máquina workstation usando o comando curl.

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

Você deverá ver a seguinte saída:

```
DO180 Page
```

► 3. Use o comando podman diff para examinar as diferenças no contêiner entre a imagem e a nova camada criada pelo contêiner.

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



nota

Com frequência, as imagens de contêiner do servidor web rotulam o diretório /var/www/html como um volume. Nesses casos, os arquivos adicionados a esse diretório não são considerados como parte do sistema de arquivos de contêiner e não são exibidos na saída do comando podman diff. A imagem de contêiner quay.io/redhattraining/httpd-parent não rotula o diretório /var/www/html como um volume. Portanto, a alteração no arquivo /var/www/html/do180.html é considerada uma alteração no sistema de arquivos de contêiner subjacente.

► 4. Crie uma nova imagem com as alterações criadas pelo contêiner em execução.

4.1. Pare o contêiner official-httdp.

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

- 4.2. Salve as alterações em uma nova imagem de contêiner com um novo nome. Use seu nome como o autor das alterações.

```
[student@workstation ~]$ podman commit \  
> -a 'Your Name' official-httdp do180-custom-httdp  
Getting image source signatures  
Skipping fetch of repeat blob sha256:071d8bd765171080d01682844524be57ac9883e...  
...output omitted...  
Copying blob sha256:1e19be875ce6f5b9dece378755eb9df96ee205abfb4f165c797f59a9...  
15.00 KB / 15.00 KB [=====] 0s  
Copying config sha256:8049dc2e7d0a0b1a70fc0268ad236399d9f5fb686ad4e31c7482cc...  
2.99 KB / 2.99 KB [=====] 0s  
Writing manifest to image destination  
Storing signatures  
'31c3ac78e9d4`137c928da23762e7d32b00c428eb1036cab1caeeb399befef2a23
```

- 4.3. Liste as imagens de contêiner disponíveis.

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

A saída esperada é semelhante à seguinte:

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

A ID da imagem corresponde aos 12 primeiros caracteres do hash. As imagens mais recentes são listadas no topo.

► 5. Publique a imagem de contêiner salva no registro do contêiner.

- 5.1. Carregue a configuração do ambiente de sala de aula.

Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- 5.2. Para marcar a imagem com o nome do host e a tag do registro, execute o comando a seguir.

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

- 5.3. Execute o comando `podman images` para se certificar de que o novo nome foi adicionado ao cache.

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

REPOSITORY	TAG	IMAGE ID	...
localhost/do180-custom-httdp	latest	31c3ac78e9d4	...
quay.io/\${RHT_OCP4_QUAY_USER}/do180-custom-httdp	v1.0	31c3ac78e9d4	...
quay.io/redhattraining/httdp-parent	latest	2cc07fbb5000	...

5.4. Publique a imagem no registro do Quay.io.

```
[student@workstation ~]$ podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
Getting image source signatures
Copying blob sha256:071d8bd765171080d01682844524be57ac9883e53079b6ac66707e19...
200.44 MB / 200.44 MB [=====] 1m38s
...output omitted...
Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeab3...
2.99 KB / 2.99 KB [=====] 0s
Writing manifest to image destination
Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeab3...
0 B / 2.99 KB [-----] 0s
Writing manifest to image destination
Storing signatures
```



nota

Enviar a imagem `do180-custom-httdp` cria um repositório privado na sua conta do Quay.io. Atualmente, repositórios privados são não permitidos por planos gratuitos do Quay.io. Você pode criar o repositório público antes de enviar a imagem ou alterar o repositório para público posteriormente.

5.5. Verifique se a imagem está disponível no Quay.io. O comando `podman search` exige que a imagem seja indexada pelo Quay.io. Isso pode levar algumas horas, portanto, use o comando `podman pull` para buscar a imagem. Isso prova que a imagem está disponível.

```
[student@workstation ~]$ podman pull \
> -q quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeab3
```

► 6. Crie um contêiner de uma imagem recém-publicada.

Use o comando `podman run` para iniciar um novo contêiner. Use `your_quay_username/do180-custom-httdp:v1.0` como a imagem base.

```
[student@workstation ~]$ podman run -d --name test-httdp -p 8280:80 \
> ${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
c0f04e906bb12bd0e514cbd0e581d2746e04e44a468dfbc85bc29ffcc5acd16c
```

► 7. Use o comando `curl` para acessar a página HTML. Certifique-se de usar a porta 8280.

Isso deve exibir a página HTML criada na etapa anterior.

```
[student@workstation ~]$ curl http://localhost:8280/do180.html  
DO180 Page
```

Encerramento

Na workstation, execute o script `lab image-operations finish` para concluir esse laboratório.

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

Isso conclui o exercício orientado.

► Laboratório Aberto

Gerenciamento de imagens

Resultados

Você deverá ser capaz de criar uma imagem personalizada de contêiner e gerenciar imagens de contêiner.

Antes De Começar

Abra um terminal na **workstation** como usuário **student** e execute o seguinte comando:

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

Instruções

1. Use o comando `podman pull` para fazer download da imagem de contêiner `quay.io/redhattraining/nginx:1.17`. Essa imagem é uma cópia da imagem de contêiner oficial disponível em `docker.io/library/nginx:1.17`.
Certifique-se de que você fez download da imagem com êxito.
2. Inicie um novo contêiner usando a imagem do Nginx de acordo com as especificações listadas na lista a seguir.
 - *Nome:* `official-nginx`
 - *Executar como daemon:* sim
 - *Imagen do contêiner:* `nginx`
 - *Encaminhamento de porta:* da porta de host 8080 para a porta de contêiner 80.
3. Faça login no container usando o comando `podman exec`. Substitua o conteúdo do arquivo `index.html` por `D0180`. O diretório do servidor web está localizado em `/usr/share/nginx/html`.
Depois que o arquivo for atualizado, saia do contêiner e use o comando `curl` para acessar a página da web.
4. Pare o contêiner em execução e faça commit das alterações para criar uma nova imagem de contêiner. Dê à nova imagem o nome `do180/mynginx` e uma tag `v1.0-SNAPSHOT`. Use as seguintes especificações:
 - Nome da imagem: `do180/mynginx`
 - Tag da imagem: `v1.0-SNAPSHOT`
 - Author name: *seu nome*
5. Inicie um novo contêiner usando a imagem atualizada do Nginx de acordo com as especificações listadas na lista a seguir.
 - *Nome:* `official-nginx-dev`

- Executar como daemon: sim
 - Imagem do contêiner: do180/mynginx:v1.0-SNAPSHOT
 - Encaminhamento de porta: da porta de host 8080 para a porta de contêiner 80.
6. Faça login no contêiner usando o comando `podman exec` e faça uma alteração final. Substitua o conteúdo do arquivo `/usr/share/nginx/html/index.html` por D0180 Page.
- Depois que o arquivo for atualizado, saia do contêiner e use o comando `curl` para verificar as alterações.
7. Interrompa o contêiner em execução e confirme as alterações para criar imagem de contêiner final. Dê à nova imagem o nome `do180/mynginx` e uma tag `v1.0`. Use as seguintes especificações:
- Nome da imagem: `do180/mynginx`
 - Tag da imagem: `v1.0`
 - Author name: *seu nome*
8. Remova a imagem de desenvolvimento `do180/mynginx:v1.0-SNAPSHOT` do armazenamento local de imagens.
9. Use a imagem com tag `do180/mynginx:v1.0` para criar um novo contêiner com as seguintes especificações:
- Nome do contêiner: `my-nginx`
 - Executar como daemon: sim
 - Imagem do contêiner: `do180/mynginx:v1.0`
 - Encaminhamento de porta: da porta de host 8280 para a porta de contêiner 80

Na `workstation`, use o comando `curl` para acessar o servidor web, acessível na porta 8280.

Avaliação

Classifique seu trabalho executando o comando `lab image-review grade` na sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `workstation`, execute o comando `lab image-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

► Solução

Gerenciamento de imagens

Resultados

Você deverá ser capaz de criar uma imagem personalizada de contêiner e gerenciar imagens de contêiner.

Antes De Começar

Abra um terminal na `workstation` como usuário `student` e execute o seguinte comando:

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

Instruções

1. Use o comando `podman pull` para fazer download da imagem de contêiner `quay.io/redhattraining/nginx:1.17`. Essa imagem é uma cópia da imagem de contêiner oficial disponível em `docker.io/library/nginx:1.17`. Certifique-se de que você fez download da imagem com êxito.

- 1.1. Use o comando `podman pull` para fazer pull da imagem de contêiner do Nginx.

```
[student@workstation ~]$ podman pull quay.io/redhattraining/nginx:1.17
Trying to pull quay.io/redhattraining/nginx:1.17...
...output omitted...
Storing signatures
9beeba249f3ee158d3e495a6ac25c5667ae2de8a43ac2a8bfd2bf687a58c06c9
```

- 1.2. Certifique-se de que a imagem de contêiner exista no sistema local executando o comando `podman images`.

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

Esse comando produz uma saída semelhante à seguinte:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
quay.io/redhattraining/nginx	1.17	9beeba249f3e	6 months ago	428MB

2. Inicie um novo contêiner usando a imagem do Nginx de acordo com as especificações listadas na lista a seguir.
 - *Nome:* `official-nginx`
 - *Executar como daemon:* sim
 - *Imagem do contêiner:* `nginx`
 - *Encaminhamento de porta:* da porta de host 8080 para a porta de contêiner 80.

- 2.1. Na workstation, use o comando `podman run` para criar um contêiner chamado `official-nginx`.

```
[student@workstation ~]$ podman run --name official-nginx \
> -d -p 8080:80 quay.io/redhattraining/nginx:1.17
b9d5739af239914b371025c38340352ac1657358561e7ebbd5472dfd5ff97788
```

3. Faça login no container usando o comando `podman exec`. Substitua o conteúdo do arquivo `index.html` por D0180. O diretório do servidor web está localizado em `/usr/share/nginx/html`.

Depois que o arquivo for atualizado, saia do contêiner e use o comando `curl` para acessar a página da web.

- 3.1. Faça login no container usando o comando `podman exec`.

```
[student@workstation ~]$ podman exec -it official-nginx /bin/bash
root@b9d5739af239:/#
```

- 3.2. Atualize o arquivo `index.html` localizado em `/usr/share/nginx/html`. O arquivo deve mostrar D0180.

```
root@b9d5739af239:/# echo 'D0180' > /usr/share/nginx/html/index.html
```

- 3.3. Saia do contêiner.

```
root@b9d5739af239:/# exit
exit
```

- 3.4. Use o comando `curl` para se certificar de que o arquivo `index.html` esteja atualizado.

```
[student@workstation ~]$ curl 127.0.0.1:8080
D0180
```

4. Pare o contêiner em execução e faça commit das alterações para criar uma nova imagem de contêiner. Dê à nova imagem o nome `do180/mynginx` e uma tag `v1.0-SNAPSHOT`. Use as seguintes especificações:

- Nome da imagem: `do180/mynginx`
- Tag da imagem: `v1.0-SNAPSHOT`
- Author name: *seu nome*

- 4.1. Use o comando "podman stop" para interromper o contêiner `official-nginx`.

```
[student@workstation ~]$ podman stop official-nginx
official-nginx
```

- 4.2. Salve suas alterações em uma nova imagem de contêiner. Use seu nome como o autor das alterações.

```
[student@workstation ~]$ podman commit -a 'Your Name' \
> official-nginx do180/mynginx:v1.0-SNAPSHOT
Getting image source signatures
...output omitted...
Storing signatures
d6d10f52e258e4e88c181a56c51637789424e9261b208338404e82a26c960751
```

4.3. Liste as imagens de contêiner disponíveis para localizar sua imagem recém-criada.

```
[student@workstation ~]$ podman images
REPOSITORY           TAG      IMAGE ID      CREATED     ...
localhost/do180/mynginx   v1.0-SNAPSHOT  d6d10f52e258  30 seconds ago  ...
quay.io/redhattraining/nginx  1.17       9beeba249f3e  6 months ago  ...
```

5. Inicie um novo contêiner usando a imagem atualizada do Nginx de acordo com as especificações listadas na lista a seguir.

- *Nome:* official-nginx-dev
- *Executar como daemon:* sim
- *Imagen do contêiner:* do180/mynginx:v1.0-SNAPSHOT
- *Encaminhamento de porta:* da porta de host 8080 para a porta de contêiner 80.

- 5.1. Na workstation, use o comando podman run para criar um contêiner chamado official-nginx-dev.

```
[student@workstation ~]$ podman run --name official-nginx-dev \
> -d -p 8080:80 do180/mynginx:v1.0-SNAPSHOT
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a28866156671f0bbbb
```

6. Faça login no contêiner usando o comando podman exec e faça uma alteração final. Substitua o conteúdo do arquivo /usr/share/nginx/html/index.html por D0180 Page. Depois que o arquivo for atualizado, saia do contêiner e use o comando curl para verificar as alterações.

- 6.1. Faça login no container usando o comando podman exec.

```
[student@workstation ~]$ podman exec -it official-nginx-dev /bin/bash
root@cfa21f02a77d:/#
```

- 6.2. Atualize o arquivo index.html localizado em /usr/share/nginx/html. O arquivo deve mostrar D0180 Page.

```
root@cfa21f02a77d:/# echo 'D0180 Page' > /usr/share/nginx/html/index.html
```

- 6.3. Saia do contêiner.

```
root@cfa21f02a77d:/# exit  
exit
```

- 6.4. Use o comando `curl` para se certificar de que o arquivo `index.html` esteja atualizado.

```
[student@workstation ~]$ curl 127.0.0.1:8080  
D0180 Page
```

7. Interrompa o contêiner em execução e confirme as alterações para criar imagem de contêiner final. Dê à nova imagem o nome `do180/mynginx` e uma tag `v1.0`. Use as seguintes especificações:

- Nome da imagem: `do180/mynginx`
- Tag da imagem: `v1.0`
- Author name: `seu nome`

- 7.1. Use o comando "podman stop" para interromper o contêiner `official-nginx-dev`.

```
[student@workstation ~]$ podman stop official-nginx-dev  
official-nginx-dev
```

- 7.2. Salve suas alterações em uma nova imagem de contêiner. Use seu nome como o autor das alterações.

```
[student@workstation ~]$ podman commit -a 'Your Name' \  
> official-nginx-dev do180/mynginx:v1.0  
Getting image source signatures  
...output omitted...  
Storing signatures  
90915976c33de534e06778a74d2a8969c25ef5f8f58c0c1ab7aeaac19abd18af
```

- 7.3. Liste as imagens de contêiner disponíveis para localizar sua imagem recém-criada.

```
[student@workstation ~]$ podman images  
REPOSITORY          TAG      IMAGE ID      CREATED     ...  
localhost/do180/mynginx    v1.0      90915976c33d  6 seconds ago  ...  
localhost/do180/mynginx    v1.0-SNAPSHOT  d6d10f52e258  8 minutes ago  ...  
quay.io/redhattraining/nginx  1.17      9beeba249f3e  6 months ago  ...
```

8. Remova a imagem de desenvolvimento `do180/mynginx:v1.0-SNAPSHOT` do armazenamento local de imagens.

- 8.1. Apesar de ter sido interrompido, `official-nginx-dev` ainda está presente. Mostre o recipiente com o comando `podman ps` com o sinalizador `-a`.

```
[student@workstation ~]$ podman ps -a \
> --format="{{.ID}} {{.Names}} {{.Status}}"
cfa21f02a77d    official-nginx-dev   Exited (0) 9 minutes ago
b9d5739af239    official-nginx       Exited (0) 12 minutes ago
```

8.2. Remova o contêiner com o comando `podman rm`.

```
[student@workstation ~]$ podman rm official-nginx-dev
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a28866156671f0bbbb
```

8.3. Verifique se o contêiner foi excluído reenviando o mesmo comando `podman ps`.

```
[student@workstation ~]$ podman ps -a \
> --format="{{.ID}} {{.Names}} {{.Status}}"
b9d5739af239    official-nginx       Exited (0) 12 minutes ago
```

8.4. Use o comando "podman rmi" para remover a imagem do `do180/mynginx:v1.0-SNAPSHOT`.

```
[student@workstation ~]$ podman rmi do180/mynginx:v1.0-SNAPSHOT
Untagged: localhost/do180/mynginx:v1.0-SNAPSHOT
```

8.5. Verifique se a imagem não está mais presente listando todas as imagens usando o comando `podman images`.

```
[student@workstation ~]$ podman images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
localhost/do180/mynginx     v1.0    90915976c33d  5 minutes ago  131MB
quay.io/redhattraining/nginx   1.17   9beeba249f3e  6 months ago  131MB
```

9. Use a imagem com tag `do180/mynginx:v1.0` para criar um novo contêiner com as seguintes especificações:

- Nome do contêiner: `my-nginx`
- Executar como daemon: sim
- Imagem do contêiner: `do180/mynginx:v1.0`
- Encaminhamento de porta: da porta de host 8280 para a porta de contêiner 80

Na `workstation`, use o comando `curl` para acessar o servidor web, acessível na porta 8280.

9.1. Use o comando "podman run" para criar o contêiner `my-nginx` de acordo com as especificações.

```
[student@workstation ~]$ podman run -d --name my-nginx \
> -p 8280:80 do180/mynginx:v1.0
51958c8ec8d2613bd26f85194c66ca96c95d23b82c43b23b0f0fb9fded74da20
```

- 9.2. Use o comando `curl` para assegurar que a página `index.html` esteja disponível e retorne o conteúdo personalizado.

```
[student@workstation ~]$ curl 127.0.0.1:8280  
DO180 Page
```

Avaliação

Classifique seu trabalho executando o comando `lab image-review grade` na sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `workstation`, execute o comando `lab image-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

Sumário

Neste capítulo, você aprendeu que:

- O Red Hat Container Catalog oferece imagens testadas e certificadas em `registry.redhat.io`.
- O Podman pode interagir com registros de contêiner remotos para pesquisar, fazer pull e enviar imagens de contêiner.
- As tags de imagem são um mecanismo para dar suporte a várias versões de uma imagem de contêiner.
- O Podman fornece comandos para gerenciar imagens de contêiner no armazenamento local e como arquivos compactados.
- Use o comando `podman commit` para criar uma imagem a partir de um contêiner.

capítulo 5

Criação de imagens personalizadas de contêiner

Meta

Projetar e codificar um Containerfile para criar uma imagem personalizada de contêiner.

Objetivos

- Descrever as abordagens para criar imagens personalizadas de contêiner.
- Criar uma imagem de contêiner usando comandos comuns do Containerfile.

Seções

- Design para imagens personalizadas de contêiner (e teste)
- Criação de imagens personalizadas de contêiner com Containerfiles (e exercício orientado)

Laboratório

- Criação de imagens personalizadas de contêiner

Design para imagens personalizadas de contêiner

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Descrever as abordagens para criar imagens personalizadas de contêiner.
- Encontre Containerfiles existentes para usar como um ponto inicial a fim de criar uma imagem personalizada de contêiner.
- Defina a função do Red Hat Software Collections Library (RHSC) ao projetar imagens de contêiner a partir do registro da Red Hat.
- Descreva a alternativa do Source-to-Image (S2I) ao Containerfiles.

Reutilização de Containerfiles existentes

Um método de criação de imagens de contêiner discutido anteriormente exige que você crie um contêiner, modifique-o para atender aos requisitos do aplicativo a ser executado nele e confirme as alterações em uma imagem. Essa opção, embora simples, é adequada apenas para usar ou testar alterações muito específicas. Ela não segue as práticas recomendadas de software, como manutenção, automação de construção e repetibilidade.

Os Containerfiles são outra opção para criar imagens de contêiner, solucionando essas limitações. Containerfiles têm fácil compartilhamento, controle de versão, reutilização e extensão.

Eles também facilitam estender uma imagem, chamada de imagem filho, a partir de outra imagem, chamada de imagem pai. Uma imagem filho incorpora tudo que há na imagem pai e todas as alterações e adições feitas para criá-la.



nota

No restante deste curso, um arquivo chamado Containerfile pode ser um arquivo chamado Containerfile ou Dockerfile. Ambos compartilham a mesma sintaxe. Containerfile é o nome padrão usado pelas ferramentas compatíveis com OCI.

Para compartilhar e reutilizar imagens, muitos aplicativos populares, idiomas e estruturas já estão disponíveis em registros de imagens públicas, como Quay.io [<https://quay.io>]. É necessário personalizar as configurações de um aplicativo a fim de seguir as práticas recomendadas para contêineres. Então, iniciar a partir de uma imagem mãe comprovada normalmente economiza muito o trabalho.

Usar uma imagem pai de alta qualidade aumenta a manutenção, especialmente se a imagem pai for mantida atualizada por seu autor para lidar com correções de bugs e problemas de segurança.

A criação um Containerfile para construir uma imagem filha a partir de uma imagem de contêiner existente é, muitas vezes, usada nos seguintes cenários comuns:

- Adicionar novas bibliotecas de tempo de execução, como conectores de banco de dados.

- Incluir personalização em toda a organização, como certificados SSL e provedores de autenticação.
- Adicionar bibliotecas internas para serem compartilhadas como uma camada única de imagem por várias imagens de contêiner para diferentes aplicativos.

Alterar um Containerfile existente para criar uma nova imagem pode ser uma boa abordagem em outros cenários. Por exemplo:

- Diminua a imagem de contêiner, removendo materiais não usados (como páginas do man ou a documentação encontrada em /usr/share/doc).
- Bloqueie a imagem pai ou algum pacote de software incluído para uma versão específica a fim de diminuir o risco relacionado às futuras atualizações de software.

Duas origens de imagens de contêiner para usar como imagens pai ou para alterar seus Containerfiles são o Docker Hub e o Red Hat Software Collections Library (RHSCl).

Uso do Red Hat Software Collections Library

O Red Hat Software Collections Library (RHSCl) ou simplesmente Software Collections, é uma solução da Red Hat para desenvolvedores que precisam usar as ferramentas mais recentes de desenvolvimento e que, normalmente, não se encaixam no cronograma de versões padrão do RHEL.

O Red Hat Enterprise Linux (RHEL) fornece um ambiente estável para aplicativos corporativos. Isso exige que o RHEL mantenha as principais versões de pacotes upstream no mesmo nível para impedir alterações de formato de API e arquivo de configuração. É feito o backporting de correções de segurança e desempenho de versões de upstream mais recentes, mas não de novos recursos que não são compatíveis com versões anteriores.

O RHSCl permite que os desenvolvedores de software usem a versão mais recente sem afetar o RHEL, porque os pacotes RHSCl não substituem ou entram em conflito com os pacotes RHEL padrão. Os pacotes RHEL padrão e os pacotes RHSCl são instalados lado a lado.



nota

Todos os assinantes do RHEL têm acesso ao RHSCl. Para ativar uma determinada coleção de softwares para um usuário ou ambiente de aplicativo específico (por exemplo, MySQL 5.7, que é chamado `rh-mysql57`), ative os repositórios Yum de softwares RHSCl e siga algumas etapas simples.

Localização de Dockerfiles a partir do Red Hat Software Collections Library

O RHSCl é a origem da maioria das imagens de contêiner fornecidas pelo registro de imagens da Red Hat para uso de clientes do RHEL Atomic Host e do OpenShift Container Platform.

A Red Hat fornece os Containerfiles RHSCl e origens relacionadas no pacote `rhscl-dockerfiles` disponível no repositório RHSCl. Os usuários da comunidade podem obter os Containerfiles para imagens de contêiner equivalentes baseadas em CentOS do repositório GitHub em <https://github.com/sclorg?q=-container>.

**nota**

Muitas imagens de contêiner do RHSCl incluem suporte a Source-to-Image (S2I), mais conhecido como um recurso do OpenShift Container Platform. Oferecer suporte para S2I não afeta o uso dessas imagens de contêiner com o Docker.

Imagens de contêiner no Red Hat Container Catalog (RHCC)

Aplicativos críticos exigem contêineres confiáveis. O Red Hat Container Catalog é um repositório de coleções confiáveis, testadas, certificadas e organizadas de imagens de contêiner compiladas em versões do Red Hat Enterprise Linux (RHEL) e sistemas relacionados. As imagens de contêiner disponíveis por meio do RHCC foram submetidas a um processo de garantia de qualidade. Todos os componentes foram recompilados pela Red Hat para evitar riscos de segurança conhecidos. Eles são atualizados regularmente para garantir que eles contenham a versão exigida do software mesmo quando uma nova imagem ainda não está disponível. Usando o RHCC, você pode procurar imagens e acessar informações sobre elas, como a versão, o conteúdo e a utilização.

Pesquisar imagens usando o Quay.io

O Quay.io é um repositório de contêiner avançado do CoreOS otimizado para colaboração em equipe. Você pode pesquisar por imagens de contêiner usando <https://quay.io/search>.

Clicar no nome de uma imagem fornece acesso à página de informações da imagem, incluindo o acesso a todas as tags existentes para a imagem e o comando para enviar a imagem.

Localização de Containerfiles no Docker Hub

Tenha cuidado com as imagens do Docker Hub. Qualquer pessoa pode criar uma conta do Docker Hub publicar imagens de contêiner lá. Não há garantias gerais de qualidade e segurança. As imagens no Docker Hub variam entre aquelas com suporte profissional e experimentos únicos. Cada imagem pode ser avaliada individualmente.

Depois de pesquisar uma imagem, a página de documentação pode fornecer um link para seu Containerfile. Por exemplo, o primeiro resultado ao pesquisar `mysql` é a página de documentação para a imagem MySQL official em https://hub.docker.com/_/mysql.

Nessa página, na seção `Supported tags and respective Dockerfile links`, cada uma das tags aponta para o projeto GitHub `docker-library`, que hospeda Containerfiles para imagens criadas pelo sistema de criação automática da comunidade do Docker.

A URL direta para a árvore do Containerfile Docker Hub MySQL 8.0 é <https://github.com/docker-library/mysql/blob/master/8.0>.

Descrição do uso da ferramenta OpenShift Source-to-Image

O Source-to-Image (S2I) fornece uma alternativa ao uso de Containerfiles para criar novas imagens de contêiner que pode ser usado como um recurso do OpenShift ou como o utilitário autônomo `s2i`. O S2I permite que os desenvolvedores trabalhem usando suas ferramentas usuais, em vez de aprender a sintaxe do Containerfile e usar comandos do sistema operacional, como `yum`. O utilitário S2I normalmente cria imagens mais leves com menos camadas.

capítulo 5 | Criação de imagens personalizadas de contêiner

O S2I usa o seguinte processo para compilar uma imagem personalizada de contêiner para um aplicativo:

1. Inicie um contêiner a partir da imagem de contêiner de base chamada builder image. Essa imagem inclui um tempo de execução de linguagem de programação e ferramentas de desenvolvimento essenciais como compiladores e gerentes de pacotes.
2. Busque o código-fonte do aplicativo, normalmente de um servidor Git, e envie-o ao contêiner.
3. Crie os arquivos binários de aplicativo no contêiner.
4. Salve o contêiner, depois de limpar, como uma nova imagem de contêiner, que inclua o tempo de execução de linguagem de programação e os binários do aplicativo.

A imagem do construtor é uma imagem de contêiner comum que segue uma estrutura padrão de diretório e fornece scripts que são chamados durante o processo S2I. A maioria dessas imagens de construtor também podem ser usadas como imagens de base para Containerfiles fora do processo S2I.

O comando `s2i` é usado para executar o processo S2I fora do OpenShift, em um ambiente exclusivo do Docker. Ele pode ser instalado em um sistema RHEL a partir do pacote RPM `source-to-image`, e em outras plataformas, incluindo Windows e MacOS, a partir dos instaladores disponíveis no projeto S2I no GitHub.



Referências

Red Hat Software Collections Library (RHSCl)

<https://access.redhat.com/documentation/en/red-hat-software-collections/>

Red Hat Container Catalog (RHCC)

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

Dockerfiles RHSCl no GitHub

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

Uso de imagens de contêiner do Red Hat Software Collections

<https://access.redhat.com/articles/1752723>

Quay.io

<https://quay.io/search>

Docker Hub

<https://hub.docker.com/>

Projeto Docker Library GitHub

<https://github.com/docker-library>

Projeto S2I GitHub

<https://github.com/openshift/source-to-image/>

► Teste

Abordagens para projetar imagens de contêiner

Escolha as respostas corretas para as seguintes perguntas:

- ▶ 1. **Qual método é recomendado pela comunidade de contêiner para criar imagens de contêiner?**
 - a. Executar comandos em um contêiner básico de sistema operacional, confirmar o contêiner, salvá-lo e exportá-lo como uma nova imagem de contêiner.
 - b. Executar comandos em um Containerfile e enviar a imagem de contêiner gerada a um registro de imagens.
 - c. Criar as camadas de imagens de contêiner manualmente a partir de arquivos tar.
 - d. Executar o comando `podman build` para processar uma descrição de imagem de contêiner no formato YAML.

- ▶ 2. **Qual das opções a seguir é uma vantagem de usar o processo S2I autônomo como uma alternativa ao Containerfiles?**
 - a. Não exige ferramentas adicionais além de uma configuração básica do Podman.
 - b. Cria imagens menores de contêineres com menos camadas.
 - c. Reutiliza imagens de construtor de alta qualidade.
 - d. Atualiza automaticamente a imagem filha conforme a imagem mãe é alterada (por exemplo, com correções de segurança).
 - e. Cria imagens compatíveis com o OpenShift, diferente das imagens de contêiner criadas com ferramentas do Docker.

- ▶ 3. **Quais são os cenários típicos para a criação de um Containerfile para compilar uma imagem filho a partir de uma imagem existente? (Escolha duas opções.)**
 - a. Adicionar bibliotecas de tempo de execução.
 - b. Definir restrições de acesso em um contêiner à CPU do computador host.
 - c. Adicionar bibliotecas internas para serem compartilhadas como uma camada única de imagem por várias imagens de contêiner para diferentes aplicativos.

► Solução

Abordagens para projetar imagens de contêiner

Escolha as respostas corretas para as seguintes perguntas:

- 1. Qual método é recomendado pela comunidade de contêiner para criar imagens de contêiner?**
- a. Executar comandos em um contêiner básico de sistema operacional, confirmar o contêiner, salvá-lo e exportá-lo como uma nova imagem de contêiner.
 - b. Executar comandos em um Containerfile e enviar a imagem de contêiner gerada a um registro de imagens.
 - c. Criar as camadas de imagens de contêiner manualmente a partir de arquivos tar.
 - d. Executar o comando podman build para processar uma descrição de imagem de contêiner no formato YAML.
- 2. Qual das opções a seguir é uma vantagem de usar o processo S2I autônomo como uma alternativa ao Containerfiles?**
- a. Não exige ferramentas adicionais além de uma configuração básica do Podman.
 - b. Cria imagens menores de contêineres com menos camadas.
 - c. Reutiliza imagens de construtor de alta qualidade.
 - d. Atualiza automaticamente a imagem filha conforme a imagem mãe é alterada (por exemplo, com correções de segurança).
 - e. Cria imagens compatíveis com o OpenShift, diferente das imagens de contêiner criadas com ferramentas do Docker.
- 3. Quais são os cenários típicos para a criação de um Containerfile para compilar uma imagem filho a partir de uma imagem existente? (Escolha duas opções.)**
- a. Adicionar bibliotecas de tempo de execução.
 - b. Definir restrições de acesso em um contêiner à CPU do computador host.
 - c. Adicionar bibliotecas internas para serem compartilhadas como uma camada única de imagem por várias imagens de contêiner para diferentes aplicativos.

Criação de imagens personalizadas de contêiner com Containerfiles

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de criar uma imagem de contêiner usando comandos comuns do Containerfile.

Criação de contêineres de base

Um Containerfile é um mecanismo para automatização da criação de imagens de contêiner. A criação de uma imagem a partir de um Containerfile é um processo de três etapas.

1. Crie um diretório de trabalho
2. Grave o containerfile
3. Crie a imagem com o Podman

Criação de um diretório de trabalho.

O diretório de trabalho é o diretório que contém todos os arquivos necessários para criar a imagem. Criar um diretório de trabalho vazio é uma prática recomendada para evitar a incorporação de arquivos desnecessários à imagem. Por motivos de segurança, o diretório raiz, /, não deve nunca ser usado como um diretório de trabalho para criações de imagens.

Gravação da especificação do Containerfile

Um Containerfile é um arquivo de texto chamado *Containerfile* ou *Dockerfile* que contém um conjunto de instruções necessário para criar a imagem. A sintaxe básica de um Containerfile é apresentada da seguinte maneira:

```
# Comment
INSTRUCTION arguments
```

As linhas que começam com uma cerquilha (#) são comentários de estados *INSTRUCTION* para qualquer palavra-chave de instrução do Containerfile. As instruções não diferenciam maiúsculas e minúsculas, mas a convenção comum é fazer instruções em maiúsculas para melhorar a visibilidade.

A primeira instrução que não é um comentário deve ser uma instrução *FROM* para especificar a imagem de base. As instruções do Containerfile são executadas em um novo contêiner usando essa imagem e, em seguida, confirmadas em uma nova imagem. A próxima instrução (se houver) é executada nessa nova imagem. A ordem de execução das instruções é a igual à ordem em que aparecem no Containerfile.



nota

A instrução *ARG* pode aparecer antes da instrução *FROM*, mas instruções *ARG* estão fora dos objetivos desta seção.

Cada instrução Containerfile é executada em um contêiner independente usando uma imagem intermediária criada a partir de todos os comandos anteriores. Isso significa que cada instrução é independente de outras instruções no Containerfile. Veja a seguir um Containerfile de exemplo para criar um contêiner simples de servidor da web do Apache:

```
# This is a comment line ①
FROM ubi8/ubi:8.5 ②
LABEL description="This is a custom httpd container image" ③
MAINTAINER John Doe <jdoe@xyz.com> ④
RUN yum install -y httpd ⑤
EXPOSE 80 ⑥
ENV LogLevel "info" ⑦
ADD http://someserver.com/filename.pdf /var/www/html ⑧
COPY ./src/ /var/www/html/ ⑨
USER apache ⑩
ENTRYPOINT ["/usr/sbin/httpd"] ⑪
CMD ["-D", "FOREGROUND"] ⑫
```

- ① As linhas que começam com uma cerquilha (#) são comentários.
- ② A instrução FROM declara que a nova imagem de container estende a imagem base de contêiner `ubi8/ubi:8.5`. Os Containerfiles podem usar qualquer outra imagem de contêiner como a imagem de base, não apenas imagens das distribuições do sistema operacional. A Red Hat fornece um conjunto de imagens de contêiner que são certificadas e testadas, e recomenda usar essas imagens de contêiner como base.
- ③ LABEL é responsável por adicionar metadados genéricos a uma imagem. Um LABEL é par simples de chave-valo.
- ④ MAINTAINER indica o campo Author dos metadados da imagem do contêiner gerado. Você pode usar o comando `podman inspect` para exibir os metadados de imagem.
- ⑤ RUN executa comandos em uma nova camada acima da imagem atual. O shell usado para executar comandos é `/bin/sh`.
- ⑥ EXPOSE indica que o contêiner escuta na porta de rede especificada no tempo de execução. A instrução EXPOSE apenas define os metadados, ela não torna as portas acessíveis a partir do host. A opção `-p` no comando `podman run` expõe as portas do contêiner do host.
- ⑦ ENV é responsável por definir as variáveis de ambiente que estão disponíveis no contêiner. Você pode declarar várias instruções ENV no Containerfile. Você pode usar o comando `env` no contêiner para exibir cada uma das variáveis de ambiente.
- ⑧ A instrução ADD copia arquivos ou pastas de uma origem local ou remota e os adiciona ao sistema de arquivos do contêiner. Se usado para copiar arquivos locais, eles devem estar no diretório de trabalho. A instrução ADD descompacta os arquivos `.tar` locais para o diretório de imagens de destino.
- ⑨ COPY copia arquivos do diretório de trabalho e os adiciona ao sistema de arquivos do contêiner. Não é possível copiar um arquivo remoto usando sua URL com essa instrução do Containerfile.
- ⑩ USER especifica o nome de usuário e a UID a ser usada ao executar a imagem de contêiner para as instruções RUN, CMD e ENTRYPOINT. É uma boa prática definir um usuário diferente de `root` por motivos de segurança.

capítulo 5 | Criação de imagens personalizadas de contêiner

- ⑪ ENTRYPPOINT especifica o comando padrão a executar quando a imagem é executada em um contêiner. Se omitido, o padrão ENTRYPPOINT será /bin/sh -c.
- ⑫ CMD oferece os argumentos padrões para a instrução ENTRYPPOINT. Se o padrão ENTRYPPOINT aplica (/bin/sh -c), o CMD forma um comando executável e parâmetros que são executados no início do contêiner.

CMD e ENTRYPPOINT

As instruções ENTRYPPOINT e CMD têm dois formatos:

- Formato Exec (usando um array JSON):

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

- Formato Shell:

```
ENTRYPPOINT command param1 param2
CMD param1 param2
```

Formato Exec é o formato preferido. Formato Shell envolve os comandos em um shell /bin/sh -c, criando um processo de shell que às vezes é desnecessário. Além disso, algumas combinações não são permitidas ou podem não funcionar conforme o esperado. Por exemplo, se ENTRYPPOINT é ["ping"] (formato exec) e CMD é localhost (formato shell), o comando executado esperado é ping localhost, mas o contêiner tenta ping /bin/sh -c localhost, que é um comando incorreto.

O Containerfile deve conter, no máximo, uma instrução ENTRYPPOINT e uma instrução CMD. Se mais de um de cada estiver presente, somente a última instrução entrará em vigor. O CMD pode estar presente sem especificar um ENTRYPPOINT. Nesse caso, a imagem de base ENTRYPPOINT se aplica, ou o padrão ENTRYPPOINT>se nenhum estiver definido.

O Podman pode substituir a instrução CMD ao iniciar um contêiner. Se presente, todos os parâmetros para o comando podman run após o nome da imagem formam a instrução CMD. Por exemplo, a seguinte instrução faz com que qualquer contêiner seja executado para exibir a hora atual.

```
ENTRYPPOINT ["/bin/date", "+%H:%M"]
```

O ENTRYPPOINT define o comando a ser executado e os parâmetros. Portanto, a instrução CMD não pode ser usada. O exemplo a seguir fornece a mesma funcionalidade do anterior, com o benefício da instrução CMD ser substituível quando um contêiner for iniciado.

```
ENTRYPPOINT ["/bin/date"]
CMD ["+%H:%M"]
```

Em ambos os casos, quando um contêiner é iniciado sem fornecer um parâmetro, a hora atual é exibida:

```
[student@workstation ~]$ sudo podman run -it do180/rhel
11:41
```

capítulo 5 | Criação de imagens personalizadas de contêiner

No segundo caso, se um parâmetro aparecer após o nome da imagem no comando `podman run`, ele substitui a instrução CMD. O seguinte comando exibe o dia atual da semana em vez do horário:

```
[student@workstation demo-basic]$ sudo podman run -it do180/rhel +%A  
Tuesday
```

Outra abordagem é usar o padrão ENTRYPPOINT e a instrução CMD para definir o comando inicial. A instrução a seguir exibe a hora atual, com o benefício adicional de poder ser substituída no tempo de execução.

```
CMD ["date", "+%H:%M"]
```

ADD e COPY

As instruções ADD e COPY têm dois formatos:

- O formato Shell:

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

- O formato Exec:

```
ADD ["<_source_ >", ... "<_destination_ >"]  
COPY ["<_source_ >", ... "<_destination_>"]
```

Se o *source* for um caminho do sistema de arquivos, ele deve estar dentro do diretório de trabalho.

A instrução ADD também permite que você especifique um recurso usando uma URL.

```
ADD http://someserver.com/filename.pdf /var/www/html
```

Se a *origem* for um arquivo tar local em um formato de compactação reconhecido (identity, gzip, bzip2 ou xz), a instrução ADD desempacota o arquivo como um diretório para a pasta de *destino*. A instrução COPY não tem esse recurso.



Cuidado

As instruções ADD e COPY copiam os arquivos, mantendo as permissões, com root como o proprietário, mesmo que a instrução USER seja especificada. A Red Hat recomenda que você use uma instrução RUN após a cópia para alterar o proprietário e evitar erros de permissão negada.

Imagen em camadas

Cada instrução em um Containerfile cria uma nova camada de imagem. Ter instruções demais em um Containerfile resulta em camadas em excesso, o que resulta em imagens grandes. Por exemplo, considere as seguintes instruções RUN em um Containerfile:

capítulo 5 | Criação de imagens personalizadas de contêiner

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

Este exemplo mostra a criação de uma imagem de contêiner com três camadas (uma para cada instrução RUN). A Red Hat recomenda minimizar o número de camadas. Você pode atingir o mesmo objetivo criando uma image de camada única usando a conjunção`&&` na instrução RUN.

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

O problema com essa abordagem é que a legibilidade do Containerfile decai. Use o código de escape`\` para inserir quebras de linha e melhorar a legibilidade. Você também pode recuar linhas para alinhar os comandos:

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

Esse exemplo cria apenas uma camada e a legibilidade melhora. As instruções RUN, COPY e ADD criam novas camadas de imagem, mas RUN pode ser melhorado dessa maneira.

A Red Hat recomenda a aplicação de regras de formatação semelhantes a outras instruções que aceitam vários parâmetros, como LABEL e ENV:

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

```
ENV MYSQL_ROOT_PASSWORD="my_password" \  
MYSQL_DATABASE "my_database"
```

Criação de imagens com o Podman

O comando `podman build` processa o Containerfile e cria uma nova imagem baseada nas instruções que ele contém. A sintaxe para esse comando é a seguinte:

```
$ podman build -t NAME:TAG DIR
```

DIR é o caminho do diretório de trabalho. Ele pode ser o diretório atual como designado por um ponto`(.)` se o diretório de trabalho for o diretório atual. *NAME:TAG* é um nome com uma tag atribuída à nova imagem. Se *TAG* não for especificado, a imagem será automaticamente marcada com `latest`.



nota

O diretório de trabalho atual é, por padrão, o caminho para o Containerfile, mas você pode especificar um diretório diferente usando o sinalizador -f. Para obter mais informações, você pode verificar Práticas recomendadas para gravar Dockerfiles [https://docs.docker.com/develop/develop-images/dockerfile_best-practices/].



Referências

Guia de referência do Dockerfile

<https://docs.docker.com/engine/reference/builder/>

Criação de imagens de base

<https://docs.docker.com/engine/userguide/eng-image/baseimages/>

► Exercício Guiado

Criação de uma imagem básica de contêiner do Apache

Neste exercício, você criará uma imagem básica de contêiner do Apache.

Resultados

Você deverá ser capaz de criar uma imagem de contêiner do Apache personalizada criada em uma imagem Red Hat Universal Base Image 8.

Antes De Começar

Execute o comando a seguir para fazer o download dos arquivos do laboratório relevantes e verificar se o Docker está em execução:

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

Instruções

► 1. Crie o Containerfile Apache.

- 1.1. Abra um terminal na `workstation`. Usando o editor de sua preferência, crie um novo Containerfile.

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

- 1.2. Use UBI 8.5 como uma imagem de base, adicionando a seguinte instrução `FROM` na parte superior do novo Containerfile:

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

- 1.3. Abaixo da instrução `FROM`, inclui a instrução `MAINTAINER` para definir o campo `Author` em uma nova imagem. Substitua os valores para incluir seu nome e endereço de e-mail.

```
MAINTAINER Your Name <_youremail_>
```

- 1.4. Abaixo da instrução `MAINTAINER`, adicione a seguinte instrução `LABEL` para adicionar metadados de descrição na nova imagem:

```
LABEL description="A custom Apache container based on UBI 8"
```

- 1.5. Adicione uma instrução `RUN` com um comando `yum install` para instalar o Apache no novo contêiner.

capítulo 5 | Criação de imagens personalizadas de contêiner

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

- 1.6. Adicione uma instrução RUN para substituir o conteúdo da home page do HTTPD padrão.

```
RUN echo "Hello from Containerfile" > /var/www/html/index.html
```

- 1.7. Use a instrução EXPOSE abaixo da instrução RUN para documentar a porta que o contêiner escuta no tempo de execução. Nessa instância, defina a porta como 80 porque é o padrão para um servidor Apache.

```
EXPOSE 80
```

**nota**

A instrução EXPOSE não torna a porta especificada disponível para o host, mas a instrução funciona como metadados que as portas do contêiner escutam.

- 1.8. No final do arquivo, use a seguinte instrução CMD para definir httpd como o ponto de entrada padrão:

```
CMD ["httpd", "-D", "FOREGROUND"]
```

- 1.9. Verifique se o Containerfile corresponde às seguintes informações antes de salvar ou proceder com as próximas etapas:

```
FROM ubi8/ubi:8.5

MAINTAINER Your Name <_youremail_>

LABEL description="A custom Apache container based on UBI 8"

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

RUN echo "Hello from Containerfile" > /var/www/html/index.html

EXPOSE 80

CMD ["httpd", "-D", "FOREGROUND"]
```

- 2. Compile e verifique a imagem de contêiner do Apache.

- 2.1. Use os seguintes comandos para criar uma imagem básica de contêiner do Apache usando o recém-criado Containerfile:

```
[student@workstation ~]$ cd /home/student/D0180/labs/dockerfile-create
[student@workstation dockerfile-create]$ podman build --layers=false \
> -t do180/apache .
```

capítulo 5 | Criação de imagens personalizadas de contêiner

```

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

```

- ① A imagem de contêiner listada na instrução FROM só é baixada se ainda não estiver presente no armazenamento local.

**nota**

O Podman cria muitas imagens intermediárias anônimas durante o processo de criação. Elas não devem ser listadas, a menos que -a seja usado. Use a opção --layers=false do subcomando build para instruir o Podman a excluir imagens intermediárias.

- 2.2. Quando o processo de criação for encerrado, execute podman images para ver a nova imagem no repositório de imagens.

```
[student@workstation dockerfile-create]$ $ podman images
REPOSITORY          TAG      IMAGE ID      CREATED
SIZE
localhost/do180/apache    latest   8ebfe343e08c  15 seconds ago  234
MB
registry.access.redhat.com/ubi8/ubi  8.5     4199acc83c6a  6 weeks ago   213
MB
```

► 3. Execute o contêiner Apache

- 3.1. Use o seguinte comando para executar um contêiner usando a imagem do Apache:

```
[student@workstation dockerfile-create]$ podman run --name lab-apache \
> -d -p 10080:80 do180/apache
fa1d1c450e8892ae085dd8bbf763edac92c41e6ffaa7ad6ec6388466809bb391
```

- 3.2. Execute o comando podman ps para ver o contêiner em execução.

```
[student@workstation dockerfile-create]$ podman ps
CONTAINER ID IMAGE COMMAND ...output omitted...
fa1d1c450e88 localhost/do180/apache:latest httpd -D FOREGROU...output omitted...
```

3.3. Use o comando `curl` para verificar se o servidor está em execução.

```
[student@workstation dockerfile-create]$ curl 127.0.0.1:10080
Hello from Containerfile
```

Encerramento

Na workstation, execute o script `lab dockerfile-create finish` para concluir esse laboratório.

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

Isso conclui o exercício orientado.

► Laboratório Aberto

Criação de imagens personalizadas de contêiner

Neste laboratório, você criará um Containerfile para criar uma imagem de contêiner personalizada do servidor web Apache. A imagem personalizada será baseada em uma imagem UBI RHEL 8.3 e fornecerá uma página `index.html` personalizada.

Resultados

Você deverá ser capaz de criar um contêiner personalizado de servidor web do Apache que hospeda arquivos HTML estáticos.

Antes De Começar

Abra um terminal na workstation como usuário `student` e execute o seguinte comando:

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

Instruções

- Revise o stub de Containerfile fornecido na pasta `/home/student/D0180/labs/dockerfile-review/`. Edite o `Containerfile` e garanta que ele cumpra as seguintes especificações:
 - A imagem de base é `ubi8/ubi:8.5`
 - Define o nome de autor e a ID de e-mail desejados com a instrução `Maintainer`.
 - Define a variável de ambiente `PORT` como `8080`
 - Instale o Apache (pacote `httpd`).
 - Change the Apache configuration file `/etc/httpd/conf/httpd.conf` to listen to port `8080` instead of the default port `80`.
 - Altere a propriedade das pastas `/etc/httpd/logs` e `/run/httpd` para usuário e grupo apache (UID e GID são `48`).
 - Para que os usuários de contêiner saibam como acessar o servidor web Apache, exponha o valor definido na variável de ambiente `PORT`.
 - Copie o conteúdo da pasta `src/` no diretório do laboratório para o arquivo `DocumentRoot` do Apache (`/var/www/html/`) no contêiner.

A pasta `src` contém um único arquivo `index.html` que imprime uma mensagem `Hello World!`.

 - Inicie o daemon `httpd` do Apache em primeiro plano usando a instrução `CMD` e o seguinte comando:

```
httpd -D FOREGROUND
```

2. Crie a imagem personalizada do Apache com o nome `do180/custom-apache`.
3. Crie um novo contêiner no modo desanexado com as seguintes características:
 - Nome: `containerfile`
 - Imagem do contêiner: `do180/custom-apache`
 - Encaminhamento de porta: da porta de host 20080 para a porta de contêiner 8080
 - Executar como um daemon: simVerifique se o contêiner está pronto e em execução.
4. Verifique se o servidor está fornecendo o arquivo HTML.

Avaliação

Avalie seu trabalho executando o comando `lab dockerfile-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `workstation`, execute o comando `lab dockerfile-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

► Solução

Criação de imagens personalizadas de contêiner

Neste laboratório, você criará um Containerfile para criar uma imagem de contêiner personalizada do servidor web Apache. A imagem personalizada será baseada em uma imagem UBI RHEL 8.3 e fornecerá uma página `index.html` personalizada.

Resultados

Você deverá ser capaz de criar um contêiner personalizado de servidor web do Apache que hospeda arquivos HTML estáticos.

Antes De Começar

Abra um terminal na `workstation` como usuário `student` e execute o seguinte comando:

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

Instruções

- Revise o stub de Containerfile fornecido na pasta `/home/student/D0180/labs/dockerfile-review/`. Edite o `Containerfile` e garanta que ele cumpra as seguintes especificações:
 - A imagem de base é `ubi8/ubi:8.5`
 - Define o nome de autor e a ID de e-mail desejados com a instrução `Maintainer`.
 - Define a variável de ambiente `PORT` como `8080`
 - Instale o Apache (pacote `httpd`).
 - Change the Apache configuration file `/etc/httpd/conf/httpd.conf` to listen to port `8080` instead of the default port `80`.
 - Altere a propriedade das pastas `/etc/httpd/logs` e `/run/httpd` para usuário e grupo apache (UID e GID são `48`).
 - Para que os usuários de contêiner saibam como acessar o servidor web Apache, exponha o valor definido na variável de ambiente `PORT`.
 - Copie o conteúdo da pasta `src/` no diretório do laboratório para o arquivo `DocumentRoot` do Apache (`/var/www/html/`) no contêiner.A pasta `src` contém um único arquivo `index.html` que imprime uma mensagem `Hello World!`.
- Inicie o daemon `httpd` do Apache em primeiro plano usando a instrução `CMD` e o seguinte comando:

```
httpd -D FOREGROUND
```

1. Use seu editor preferido para modificar o Containerfile localizado na pasta /home/student/D0180/labs/dockerfile-review/.

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

- 1.2. Defina a imagem base do Containerfile como ubi8/ubi:8.5.

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

- 1.3. Defina seu nome e e-mail com uma instrução MAINTAINER.

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

- 1.4. Crie uma variável de ambiente chamada PORT e a defina como 8080.

```
ENV PORT 8080
```

- 1.5. Instale o servidor do Apache.

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

- 1.6. Altere o arquivo de configuração do servidor HTTP do Apache para escutar na porta 8080 e altere a propriedade das pastas de trabalho do servidor com uma única instrução RUN.

```
RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" /etc/httpd/conf/httpd.conf && \  
    chown -R apache:apache /etc/httpd/logs/ && \  
    chown -R apache:apache /run/httpd/
```

- 1.7. Use a instrução USER para executar o contêiner como o usuário apache. Use a instrução EXPOSE para documentar a porta que o contêiner escuta no tempo de execução. Nessa instância, defina a porta para a variável de ambiente PORT, que é o padrão para um servidor Apache.

```
USER apache  
  
# Expose the custom port that you provided in the ENV var  
EXPOSE ${PORT}
```

- 1.8. Copie todos os arquivos da pasta src para o caminho DocumentRoot do Apache em /var/www/html.

```
# Copy all files under src/ folder to Apache DocumentRoot (/var/www/html)  
COPY ./src/ /var/www/html/
```

capítulo 5 | Criação de imagens personalizadas de contêiner

- 1.9. Por fim, insira uma instrução CMD para executar httpd em primeiro plano e salve o Containerfile.

```
# Start Apache in the foreground
CMD ["httpd", "-D", "FOREGROUND"]
```

2. Crie a imagem personalizada do Apache com o nome do180/custom-apache.

- 2.1. Verifique o Containerfile para a imagem personalizada do Apache.

O Containerfile para a imagem personalizada do Apache deve ser assim:

```
FROM ubi8/ubi:8.5

MAINTAINER Your Name <youremail>

ENV PORT 8080

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

RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" /etc/httpd/conf/httpd.conf && \
    chown -R apache:apache /etc/httpd/logs/ && \
    chown -R apache:apache /run/httpd/

USER apache

# Expose the custom port that you provided in the ENV var
EXPOSE ${PORT}

# Copy all files under src/ folder to Apache DocumentRoot (/var/www/html)
COPY ./src/ /var/www/html/

# Start Apache in the foreground
CMD ["httpd", "-D", "FOREGROUND"]
```

- 2.2. Execute um comando podman build para compilar a imagem personalizada do Apache com o nome do180/custom-apache.

```
[student@workstation dockerfile-review]$ podman build --layers=false \
> -t do180/custom-apache .
STEP 1: FROM ubi8/ubi:8.5
...output omitted...
STEP 2: MAINTAINER username <username@example.com>
STEP 3: ENV PORT 8080
STEP 4: RUN yum install -y httpd &&      yum clean all
...output omitted...
STEP 5: RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" /etc/httpd/conf/httpd.conf
&&      chown -R apache:apache /etc/httpd/logs/ &&      chown -R apache:apache /
run/httpd/
STEP 6: USER apache
STEP 7: EXPOSE ${PORT}
STEP 8: COPY ./src/ /var/www/html/
```

capítulo 5 | Criação de imagens personalizadas de contêiner

```
STEP 9: CMD ["httpd", "-D", "FOREGROUND"]
STEP 10: COMMIT ...output omitted... localhost/do180/custom-apache:latest
...output omitted...
```

- 2.3. Execute o comando `podman images` para verificar se a imagem personalizada foi compilada com êxito.

```
[student@workstation dockerfile-review]$ podman images
REPOSITORY                                TAG      IMAGE ID      CREATED
SIZE
localhost/do180/custom-apache             latest   08fcf6d92b16  3 minutes ago
234 MB
registry.access.redhat.com/ubi8/ubi       8.3     4199acc83c6a  6 weeks ago
213 MB
```

3. Crie um novo contêiner no modo desanexado com as seguintes características:

- Nome: `containerfile`
- Imagem do contêiner: `do180/custom-apache`
- Encaminhamento de porta: da porta de host 20080 para a porta de contêiner 8080
- Executar como um daemon: sim

Verifique se o contêiner está pronto e em execução.

- 3.1. Crie e execute o contêiner.

```
[student@workstation dockerfile-review]$ podman run -d \
> --name containerfile -p 20080:8080 do180/custom-apache
367823e35c4a...
```

- 3.2. Verifique se o contêiner está pronto e em execução.

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

4. Verifique se o servidor está fornecendo o arquivo HTML.

- 4.1. Execute um comando `curl` em `127.0.0.1:20080`

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

A saída deve ser assim:

```
<html>
<header><title>D0180 Hello!</title></header>
<body>
  Hello World! The containerfile-review lab works!
</body>
</html>
```

Avaliação

Avalie seu trabalho executando o comando `lab dockerfile-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `workstation`, execute o comando `lab dockerfile-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

Sumário

Neste capítulo, você aprendeu que:

- Um `Containerfile` contém instruções que especificam como criar uma imagem de contêiner.
- As imagens de contêiner fornecidas pelo Red Hat Container Catalog ou Quay.io são um bom ponto de partida para criar imagens personalizadas para uma linguagem ou tecnologia específica.
- A criação de uma imagem a partir de um `Containerfile` é um processo de três etapas:
 1. Crie um diretório de trabalho.
 2. Especifique as instruções de compilação em um arquivo `Containerfile`.
 3. Crie a imagem com o comando `podman build`.
- O processo Source-to-Image (S2I) oferece uma alternativa ao `Containerfiles`. O S2I implementa um processo de criação de imagem de contêiner padronizado para tecnologias comuns a partir do código-fonte do aplicativo. Isso permite que os desenvolvedores se concentrem no desenvolvimento de aplicativos e não no desenvolvimento do `Containerfile`.

capítulo 6

Implantação de aplicativos conteinerizados no OpenShift

Meta

Implantar aplicativos de contêiner único no OpenShift Container Platform.

Objetivos

- Descrever a arquitetura do Kubernetes e do Red Hat OpenShift Container Platform.
- Criar recursos básicos do Kubernetes
- Criar uma rota para um serviço.
- Compilar um aplicativo usando o recurso Source-to-Image do OpenShift Container Platform.
- Criar um aplicativo com o console da web do OpenShift.

Seções

- Descrição da arquitetura do Kubernetes e do OpenShift (e teste)
- Criação de recursos do Kubernetes (e exercício orientado)
- Criação de rotas (e exercício orientado)
- Criação de aplicativos com o recurso de origem para imagem (e exercício orientado)
- Criação de aplicativos com o console da web do OpenShift (e exercício orientado)

Laboratório

- Implantação de aplicativos conteinerizados no OpenShift

Descrição da arquitetura do Kubernetes e do OpenShift

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Descrever a arquitetura de um cluster do Kubernetes em execução no Red Hat OpenShift Container Platform (RHOCP).
- Listar os principais tipos de recursos oferecidos pelo Kubernetes e pelo RHOCP.
- Identificar as características de rede dos contêineres, Kubernetes e RHOCP.
- Listar mecanismos para disponibilizar um pod externamente.

Kubernetes e OpenShift

Nos capítulos anteriores, vimos que o Kubernetes é um serviço de orquestração que simplifica a implantação, o gerenciamento e o dimensionamento de aplicativos conteinerizados. Uma das principais vantagens de usar o Kubernetes é que ele usa vários nós para garantir a resiliência e a escalabilidade de seus aplicativos gerenciados. O Kubernetes forma um cluster de servidores de nó que executam contêineres e são gerenciados centralmente por um conjunto de servidores de plano de controle. Um servidor pode atuar tanto como nó de plano de controle quanto como nó de computação, mas tais funções são normalmente separadas para obter maior estabilidade.

Terminologia do Kubernetes

Termo	Definição
Nó	Um servidor que hospeda aplicativos em um cluster do Kubernetes.
Plano de controle	Oferece serviços básicos de cluster, como APIs ou controladores.
Nó de computação	Esse nó executa cargas de trabalho para o cluster. Os pods de aplicativos são agendados nos nós de computação.
Recurso	Recursos são qualquer tipo de definição de componente gerenciada pelo Kubernetes. Os recursos têm a configuração do componente gerenciado (por exemplo, a função atribuída a um nó) e o estado atual do componente (por exemplo, se o nó estiver disponível).
Controlador	Um controlador é um processo do Kubernetes que observa os recursos e faz alterações tentando mover o estado atual para o estado desejado.
Rótulo	Um par de chave-valor que pode ser atribuído a qualquer recurso do Kubernetes. Os seletores usam rótulos para filtrar recursos elegíveis para agendamento e outras operações.
Namespace	Um escopo para recursos e processos do Kubernetes, para que recursos com o mesmo nome possam ser usados em diferentes limites.

**nota**

As versões mais recentes do Kubernetes implementam muitos controladores como Operadores. Operadores são componentes plug-in do Kubernetes que podem reagir a eventos de cluster e controlar o estado dos recursos. Operadores e CoreOS Operator Framework estão fora do escopo deste documento.

O Red Hat OpenShift Container Platform é um conjunto de componentes e serviços modulares desenvolvidos com base no Red Hat CoreOS e no Kubernetes. O RHOPC adiciona capacidades PaaS como gerenciamento remoto, monitoramento e auditoria, segurança aumentada, gerenciamento de ciclo de vida de aplicativo e interfaces de autosserviço para desenvolvedores.

Um cluster do OpenShift é um cluster do Kubernetes que pode ser gerenciado da mesma maneira, mas usando as ferramentas de gerenciamento fornecidas pelo OpenShift, como a interface de linha de comando ou o console da web. Isso permite fluxos de trabalho mais produtivos e facilita bastante tarefas comuns.

Terminologia do OpenShift

Termo	Definição
Nó de infraestrutura	Um servidor de nó que contém serviços de infraestrutura, como monitoramento, registro ou roteamento externo.
Console	Uma interface de usuário da web fornecida pelo cluster RHOPC que permite que desenvolvedores e administradores interajam com recursos de cluster.
Projeto	Extensão do OpenShift para os namespaces do Kubernetes. Permite a definição de controle de acesso do usuário (UAC) para os recursos.

O esquema a seguir ilustra a pilha do OpenShift Container Platform:

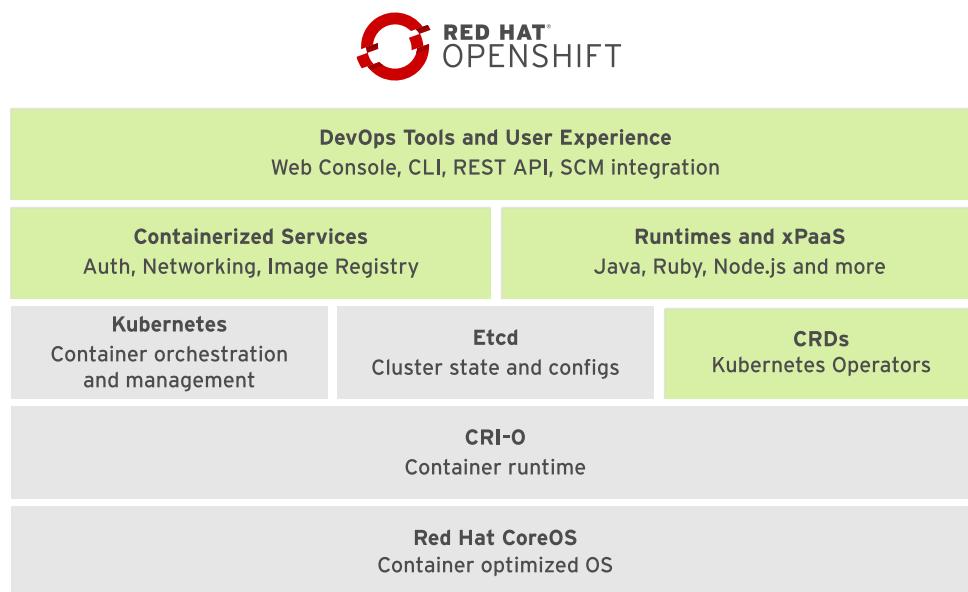


Figura 6.1: Pilha de componentes do OpenShift

De baixo para cima, da esquerda para a direita, a infraestrutura de contêiner básica é exibida, integrada e aprimorada pela Red Hat:

- O SO base é Red Hat CoreOS. O Red Hat CoreOS é uma distribuição do Linux concentrada em fornecer um sistema operacional imutável para a execução de contêineres.
- CRI-O é uma implementação do Kubernetes Container Runtime Interface (CRI) para ativar o uso de tempos de execução compatíveis com Open Container Initiative (OCI). O CRI-O pode usar qualquer tempo de execução de contêiner que satisfaça o CRI: runc (usado pelo serviço Docker), libpod (usado pelo Podman) ou rkt (do CoreOS).
- O Kubernetes gerencia um cluster de hosts (físicos ou virtuais) que executam contêineres. Ele funciona com recursos que descrevem aplicativos de vários contêineres compostos por vários recursos e como eles se interconectam.
- Etcd é um armazenamento de chave–valor distribuído, usado pelo Kubernetes para armazenar informações de configuração e estado dos contêineres e outros recursos dentro do cluster do Kubernetes.
- Definições de recursos personalizados (CRDs) são tipos de recursos armazenados no Etcd e gerenciados pelo Kubernetes. Esses tipos de recursos formam o estado e a configuração de todos os recursos gerenciados pelo OpenShift.
- Os serviços em contêiner cumprem muitas funções de infraestrutura, como de rede e autorização. O RHOCP usa a infraestrutura básica de contêiner do Kubernetes e o tempo de execução do contêiner subjacente para a maioria das funções internas. Ou seja, a maioria dos serviços internos do RHOCP funciona como contêineres orquestrados pelo Kubernetes.
- Tempos de execução e xPaaS são imagens de contêiner de base prontas para serem usadas pelos desenvolvedores, cada uma pré-configurada com um banco de dados ou linguagem de tempo de execução particular. A oferta de xPaaS é um conjunto de imagens de base para produtos Red Hat middleware, como o JBoss EAP e o ActiveMQ. Os tempos de execução de aplicativos do Red Hat OpenShift (RHOAR) são um conjunto de tempos de execução otimizados para aplicativos nativos em nuvem no OpenShift. Os tempos de execução de aplicativos disponíveis são Red Hat JBossEAP, OpenJDK, Thorntail, EclipseVert.x, SpringBoot e Node.js.
- O RHOCP oferece ferramentas de gerenciamento de interface de usuário da web e de CLI para gerenciar aplicativos de usuário e serviços do RHOCP. As ferramentas de interface de usuário da web e de CLI do OpenShift são desenvolvidas com base nas APIs REST, que podem ser usadas por ferramentas externas como IDEs e plataformas de CI.

Esta ilustração da arquitetura do OpenShift e do Kubernetes oferece mais informações sobre como os componentes de infraestrutura funcionam juntos.

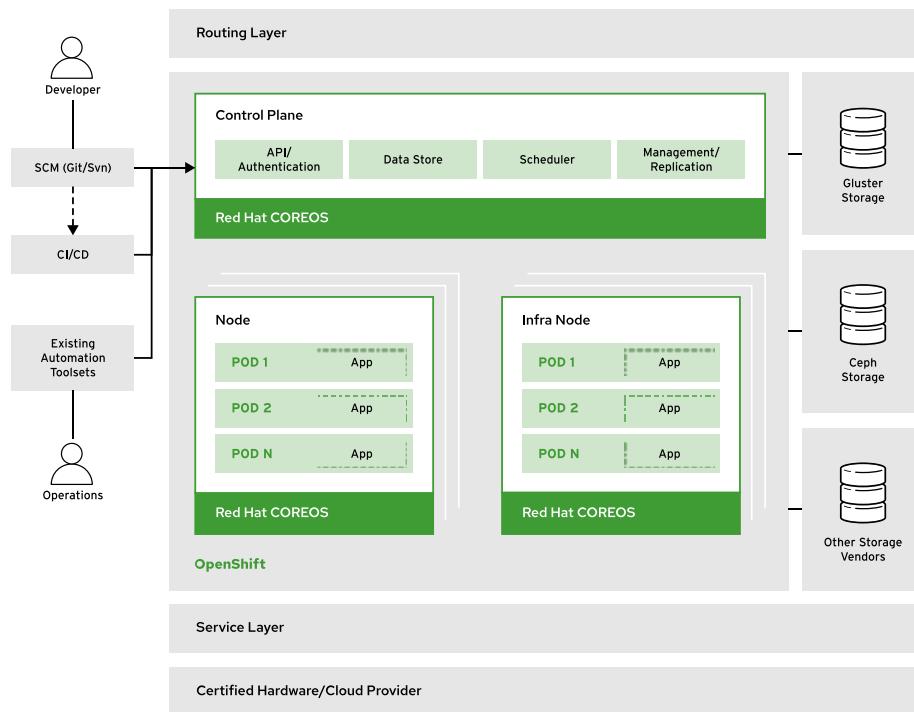


Figura 6.2: Arquitetura do OpenShift e do Kubernetes

Novos recursos no RHOC 4

O RHOC 4 é uma grande mudança em relação às versões anteriores. Além de manter compatibilidade com versões anteriores, ele inclui novos recursos, como:

- CoreOS como o sistema operacional obrigatório para todos os nós, oferecendo uma infraestrutura imutável otimizada para contêineres.
- Um novo instalador de cluster que orienta o processo de instalação e atualização.
- Uma plataforma autogerenciada, capaz de aplicar automaticamente atualizações de cluster e recuperações sem interrupções.
- Um gerenciamento de ciclo de vida do aplicativo reprojeto.
- Um SDK de operador para criar, testar e empacotar operadores.

Descrição de tipos de recurso do Kubernetes

O Kubernetes tem seis principais tipos de recursos que podem ser criados e configurados com um arquivo YAML ou JASON ou por meio das ferramentas de gerenciamento do OpenShift:

Pods (po)

Representam uma coleção de contêineres que compartilham recursos, como endereços IP e volumes de armazenamento persistentes. É a unidade básica de trabalho do Kubernetes.

Serviços (svc)

Definem uma combinação IP/porta única que fornece acesso a um pool de pods. Por padrão, os serviços conectam os clientes ao pods em regime de repetição alternada.

Controladores de replicação (rc)

Um recurso do Kubernetes que define como os pods são replicados (escalados horizontalmente) em nós diferentes. Os controladores de replicação são um serviço básico do Kubernetes para fornecer alta disponibilidade para pods e contêineres.

Volumes persistentes (pv)

Defina as áreas de armazenamento a serem usadas pelos pods do Kubernetes.

Solicitações de volume persistente (pvc)

Representam uma solicitação de armazenamento por um pod. Os PVCs vinculam um PV a um pod para que seus contêineres possam utilizá-lo, geralmente montando o armazenamento no sistema de arquivos do contêiner.

ConfigMaps (cm) e segredos

Contém um conjunto de chaves e valores que podem ser usados por outros recursos.

ConfigMaps e segredos são geralmente usados para centralizar os valores de configuração usados por vários recursos. Os segredos diferem dos mapas do ConfigMaps nos quais os valores dos segredos são sempre codificados (não criptografados) e seu acesso é restrito a menos usuários autorizados.



nota

Embora os pods do Kubernetes possam ser criados de modo independente, eles normalmente são criados por recursos de alto nível, como controladores de replicação.

Tipos de recursos do OpenShift

Os principais tipos de recurso adicionados pelo OpenShift Container Platform ao Kubernetes são apresentados a seguir:

Implantação e configuração de implantação (dc)

O OpenShift 4.5 introduziu o conceito do recurso Deployment para substituir o DeploymentConfig como a configuração padrão para pods. Ambos são a representação de um conjunto de contêineres incluídos em um pod e as estratégias de implantação a serem usadas. Ele contém a configuração a ser aplicada a todos os contêineres de cada réplica de pod, como a imagem base, tags, definições de armazenamento e os comandos a serem executados quando os contêineres iniciarem.

O objeto Deployment funciona como a versão aprimorada do objeto DeploymentConfig. Algumas substituições de funcionalidades entre ambos os objetos são as seguintes:

- A reversão automática não é mais compatível com objetos de implantação.
- Cada alteração no template de pod usado pelos objetos de implantação aciona uma nova distribuição automaticamente.
- Ganchos de ciclo de vida não são mais compatíveis com objetos de implantação.
- O processo de implantação de um objeto de implantação pode ser pausado a qualquer momento sem afetar o processo do implantador.
- Um objeto de implantação pode ter quantos conjuntos de réplicas ativos o usuário quiser, dimensionando as réplicas antigas depois de algum tempo. Em contraste, o objeto deploymentconfig só pode ter dois conjuntos de replicação ativos ao mesmo tempo.

Mesmo se os objetos Deployment devam agir como a substituição padrão de objetos DeploymentConfig, no OpenShift 4.10 os usuários ainda podem fazer uso deles se precisarem de um recurso específico fornecido por esses objetos. Nesse caso, é necessário especificar o tipo de objeto ao criar um novo aplicativo especificando o sinalizador `--as-deployment-config`.

Configuração de compilação (bc)

Define um processo a ser executado no projeto do OpenShift. Usadas pelo recurso OpenShift Source-to-Image (S2I) do OpenShift para compilar uma imagem de contêiner a partir do código-fonte de aplicativo armazenado em um repositório Git. Uma bc funciona em conjunto com uma dc para oferecer fluxos de trabalho de integração e entrega contínua básicos, porém extensíveis.

Rotas

Representam um nome de host DNS reconhecido pelo roteador OpenShift como um ponto de ingresso para aplicativos e microsserviços.



nota

Para obter uma lista de todos os recursos disponíveis em um cluster RHOPC e suas abreviações, use os comandos `oc api-resources` ou `kubectl api-resources`.

Embora os controladores de replicação do Kubernetes possam ser criados de forma independente no OpenShift, eles normalmente são criados por recursos de alto nível, como controladores de implantação.

Rede

Cada contêiner implantando em um cluster do Kubernetes tem um endereço IP atribuído de uma rede interna que é acessível somente a partir do nó executando o contêiner. Devido à natureza efêmera do contêiner, os endereços IP são constantemente atribuídos e liberados.

O Kubernetes oferece uma rede definida por software (SDN) que gera redes internas de contêiner a partir de vários nós e permite que os contêineres acessem os pods de outros hosts a partir de qualquer pod, dentro de qualquer host. O acesso à SDN só funciona de dentro do mesmo cluster do Kubernetes.

Os contêineres dentro dos pods do Kubernetes não devem se conectar aos endereços IP dinâmicos uns dos outros diretamente. Os serviços resolvem esse problema vinculando os endereços IP mais estáveis do SDN aos pods. Se os pods forem reiniciados, replicados ou reprogramados para nós diferentes, os serviços serão atualizados, fornecendo escalabilidade e tolerância a falhas.

O acesso externo a contêineres é mais complicado. Os serviços do Kubernetes podem ser definidos como um tipo de serviço NodePort. Com esse tipo de serviço, o Kubernetes aloca uma porta de rede a partir de um intervalo predefinido em cada um dos nós do cluster e a usa como proxy no seu serviço. Infelizmente, essa abordagem não apresenta um bom dimensionamento.

O OpenShift torna o acesso externo a contêineres escalável e mais simples com a definição dos recursos de rota. Uma rota define nomes DNS de interface externa e portas para um serviço. Um roteador (controlador de entrada) encaminha solicitações HTTP e TLS aos endereços de serviço dentro do SDN do Kubernetes. O único requisito é que os nomes de DNS desejados sejam mapeados para os endereços IP dos nós do roteador RHOPC.



Referências

Site de documentação do Kubernetes

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

Site de documentação do OpenShift

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

Noções básicas sobre operadores

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

► Teste

Descrição do Kubernetes e do OpenShift

Escolha as respostas corretas para as seguintes perguntas:

► 1. **Quais afirmações estão corretas em relação à arquitetura do Kubernetes? (Escolha duas opções.)**

- a. Os nós do Kubernetes podem ser gerenciados sem um plano de controle.
- b. Os planos de controle do Kubernetes gerenciam o dimensionamento de pod.
- c. Os planos de controle do Kubernetes agendam pods para nós específicos.
- d. As ferramentas do Kubernetes não podem ser usadas para gerenciar recursos em um cluster do OpenShift.
- e. Contêineres criados a partir de pods do Kubernetes não podem ser gerenciados usando ferramentas autônomas como o Podman.

► 2. **Quais duas afirmações estão corretas em relação aos tipos de recursos do Kubernetes e do OpenShift? (Escolha duas opções.)**

- a. Um pod é responsável pelo provisionamento de seu próprio armazenamento persistente.
- b. Todos os pods gerados a partir do mesmo controlador de replicação devem ser executados no mesmo nó.
- c. Um volume persistente define áreas de armazenamento disponíveis para pods por meio de uma solicitação de volume persistente.
- d. Um controlador de replicação é responsável por monitorar e manter o número de pods para um aplicativo em particular.

► 3. **Quais as duas afirmações corretas em relação à rede do Kubernetes e do OpenShift? (Escolha duas opções.)**

- a. Um serviço do Kubernetes pode fornecer um endereço IP para acessar um conjunto de pods.
- b. O Kubernetes é responsável por fornecer um nome do domínio totalmente qualificado para um pod.
- c. Um controlador de replicação é responsável por rotear solicitações externas aos pods.
- d. Uma rota é responsável por fornecer nomes de DNS para acesso externo.

► 4. Qual afirmação está correta em relação ao armazenamento persistente no OpenShift e no Kubernetes?

- a. Uma PVC representa uma área de armazenamento que um pod pode usar para armazenar dados e é provisionada pelo desenvolvedor de aplicativos.
- b. Uma PVC representa uma área de armazenamento que pode ser solicitada por um pod para armazenar dados, mas é provisionada pelo administrador do cluster.
- c. Uma PVC representa a quantidade de memória que pode ser alocada a um nó, para que um desenvolvedor possa estabelecer quanto de memória ele precisa para que seu aplicativo seja executado.
- d. Uma PVC representa o número de unidades de processamento de CPU que podem ser alocadas a um pod do aplicativo, sujeito a um limite gerenciado pelo administrador do cluster.

► 5. Qual afirmação está correta em relação às adições do OpenShift ao Kubernetes?

- a. O OpenShift adiciona recursos para simplificar a configuração do Kubernetes para muitos casos de uso reais.
- b. Imagens de contêiner criadas para o OpenShift não podem ser usadas somente com o Kubernetes.
- c. A Red Hat mantém versões bifurcadas do Kubernetes, internas ao produto do RHOC.
- d. A realização da integração e da implantação contínuas com o RHOC exige ferramentas externas.

► Solução

Descrição do Kubernetes e do OpenShift

Escolha as respostas corretas para as seguintes perguntas:

► 1. **Quais afirmações estão corretas em relação à arquitetura do Kubernetes? (Escolha duas opções.)**

- a. Os nós do Kubernetes podem ser gerenciados sem um plano de controle.
- b. Os planos de controle do Kubernetes gerenciam o dimensionamento de pod.
- c. Os planos de controle do Kubernetes agendam pods para nós específicos.
- d. As ferramentas do Kubernetes não podem ser usadas para gerenciar recursos em um cluster do OpenShift.
- e. Contêineres criados a partir de pods do Kubernetes não podem ser gerenciados usando ferramentas autônomas como o Podman.

► 2. **Quais duas afirmações estão corretas em relação aos tipos de recursos do Kubernetes e do OpenShift? (Escolha duas opções.)**

- a. Um pod é responsável pelo provisionamento de seu próprio armazenamento persistente.
- b. Todos os pods gerados a partir do mesmo controlador de replicação devem ser executados no mesmo nó.
- c. Um volume persistente define áreas de armazenamento disponíveis para pods por meio de uma solicitação de volume persistente.
- d. Um controlador de replicação é responsável por monitorar e manter o número de pods para um aplicativo em particular.

► 3. **Quais as duas afirmações corretas em relação à rede do Kubernetes e do OpenShift? (Escolha duas opções.)**

- a. Um serviço do Kubernetes pode fornecer um endereço IP para acessar um conjunto de pods.
- b. O Kubernetes é responsável por fornecer um nome do domínio totalmente qualificado para um pod.
- c. Um controlador de replicação é responsável por rotear solicitações externas aos pods.
- d. Uma rota é responsável por fornecer nomes de DNS para acesso externo.

► 4. Qual afirmação está correta em relação ao armazenamento persistente no OpenShift e no Kubernetes?

- a. Uma PVC representa uma área de armazenamento que um pod pode usar para armazenar dados e é provisionada pelo desenvolvedor de aplicativos.
- b. Uma PVC representa uma área de armazenamento que pode ser solicitada por um pod para armazenar dados, mas é provisionada pelo administrador do cluster.
- c. Uma PVC representa a quantidade de memória que pode ser alocada a um nó, para que um desenvolvedor possa estabelecer quanto de memória ele precisa para que seu aplicativo seja executado.
- d. Uma PVC representa o número de unidades de processamento de CPU que podem ser alocadas a um pod do aplicativo, sujeito a um limite gerenciado pelo administrador do cluster.

► 5. Qual afirmação está correta em relação às adições do OpenShift ao Kubernetes?

- a. O OpenShift adiciona recursos para simplificar a configuração do Kubernetes para muitos casos de uso reais.
- b. Imagens de contêiner criadas para o OpenShift não podem ser usadas somente com o Kubernetes.
- c. A Red Hat mantém versões bifurcadas do Kubernetes, internas ao produto do RHOC.
- d. A realização da integração e da implantação contínuas com o RHOC exige ferramentas externas.

Criação de recursos do Kubernetes

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de criar recursos padrão do Kubernetes.

A ferramenta de linha de comando Red Hat OpenShift Container Platform (RHOCP)

O principal método de interação com um cluster RHOCP é usar o comando `oc`. O uso básico do comando é através de seus subcomandos na seguinte sintaxe:

```
$> oc <command>
```

Antes de interagir com um cluster, a maioria das operações exige que um usuário esteja conectado. A sintaxe para fazer login é mostrada abaixo:

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

Descrição da sintaxe de definição de recurso de pod

O RHOCP executa contêineres dentro dos pods do Kubernetes e, para criar um pod a partir de uma imagem de contêiner, o OpenShift precisa de uma *definição de recurso de pod*. Isso pode ser oferecido como um arquivo de texto JSON ou YAML ou pode ser gerado a partir de padrões por meio do comando `oc new-app` ou pelo console da web do OpenShift.

Um pod é uma coleção de contêineres e outros recursos. Veja abaixo um exemplo de uma definição de pod de servidor de aplicativos WildFly em formato YAML:

```
apiVersion: v1
kind: Pod❶
metadata:
  name: wildfly❷
  labels:
    name: wildfly❸
spec:
  containers:
    - resources:
        limits:
          cpu: 0.5
      image: do276/todojee❹
      name: wildfly
      ports:
        - containerPort: 8080❺
          name: wildfly
    env:❻
      - name: MYSQL_ENV_MYSQL_DATABASE
```

```

    value: items
    - name: MYSQL_ENV_MYSQL_USER
      value: user1
    - name: MYSQL_ENV_MYSQL_PASSWORD
      value: mypa55
  
```

- ➊ Declara um tipo de recurso de pod do Kubernetes.
- ➋ Um nome único para um pod no Kubernetes que permite que administradores executem comandos nele.
- ➌ Cria um rótulo com uma chave chamado `name` que outros recursos no Kubernetes, geralmente como um serviço, podem usar para encontrá-lo.
- ➍ Define o nome da imagem de contêiner
- ➎ Um atributo dependente de contêiner identificando qual porta do contêiner está exposta.
- ➏ Define uma coleção de variáveis de ambiente.

Alguns pods podem precisar de variáveis de ambiente que possam ser lidas por um contêiner. O Kubernetes transforma todas as partes de `name` e `value` em variáveis de ambiente. Por exemplo, a variável `MYSQL_ENV_MYSQL_USER` é declarado internamente pelo tempo de execução do Kubernetes com um valor de `user1` e é encaminhado à definição de imagem de contêiner. Como o contêiner usa o mesmo nome de variável para obter o login de usuário, o valor é usado pela instância do contêiner do WildFly para definir o nome de usuário que acessa uma instância de banco de dados do MySQL.

Descrição da sintaxe de definição de recurso de serviço

O Kubernetes oferece uma rede virtual que permite que pods de diferentes nós de computação se conectem. No entanto, ele não facilita que um pod descubra os endereços IP de outros pods:

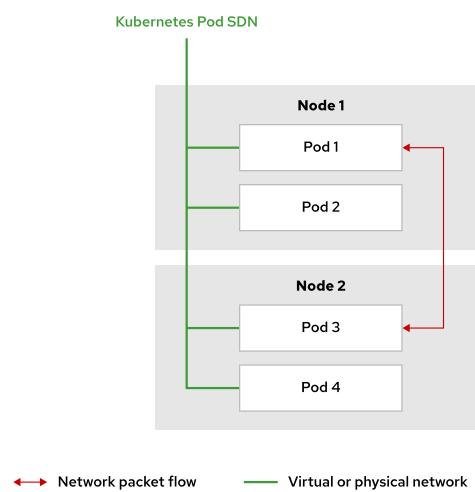


Figura 6.3: Redes básicas do Kubernetes

Se o Pod 3 falhar e for reiniciado, ele poderá retornar com um endereço IP diferente. Isso pode fazer com que o Pod 1 falhe ao tentar se comunicar com o Pod 3. Uma camada de serviço fornece a abstração necessária para resolver esse problema.

Serviços são recursos essenciais para qualquer aplicativo do OpenShift. Eles permitem que os contêineres em um pod abram conexões de rede para contêineres em outro pod. Um pod pode ser reiniciado por muitas razões, e cada vez que isso acontece, ele recebe um endereço IP interno diferente. Em vez de um pod precisar descobrir o endereço IP de outro pod após cada reinicialização, um serviço oferece um endereço IP estável para outros pods usarem, independente de qual nó de computação executa o pod após cada reinicialização.

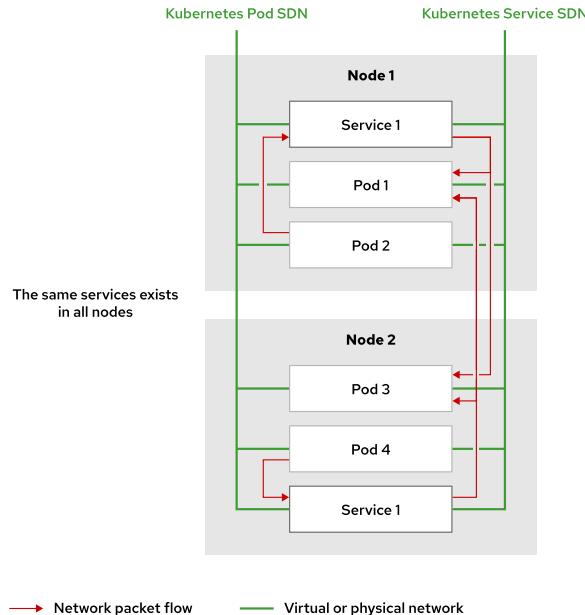


Figura 6.4: Rede de serviços do Kubernetes

A maioria dos aplicativos do mundo real não são executados como pod único. Eles precisam ser redimensionados horizontalmente, então muitos pods executam os mesmos contêineres da mesma definição de recurso de pod para atender a uma demanda crescente do usuário. Um serviço está vinculado a um conjunto de pods, oferecendo um único endereço IP para todo o conjunto e uma solicitação de cliente de平衡amento de carga entre os pods membros.

O conjunto de pods em execução por trás de um serviço é gerenciado por um recurso Deployment. Um recurso Deployment incorpora um ReplicationController que gerencia quantas cópias de pods (réplicas) devem ser criadas e cria novas cópias, caso alguma delas falhe. Os recursos Deployment e ReplicationController são explicados posteriormente neste capítulo.

O seguinte exemplo mostra uma definição de serviço mínima na sintaxe JSON:

```
{
  "kind": "Service", ①
  "apiVersion": "v1",
  "metadata": {
    "name": "quotedb" ②
  },
  "spec": {
    "ports": [ ③
      {
        "port": 3306,
        "targetPort": 3306
      }
    ]
  }
}
```

```

        ],
        "selector": {
            "name": "mysqlDb" ④
        }
    }
}

```

- ①** O tipo de recurso do Kubernetes. Neste caso, um Service.
- ②** Um nome único para o serviço.
- ③** ports é uma série de objetos que descrevem portas de rede proporcionadas pelo serviço. O atributo targetPort deve corresponder a um containerPort de uma definição de contêiner de pod, e o atributo port é a porta que é exposta pelo serviço. Os clientes se conectam à porta de serviço e o serviço encaminha pacotes ao pod targetPort.
- ④** selector é como o serviço encontra pods aos quais ele encaminha pacotes. Os pods de destino precisam ter rótulos correspondentes em seus atributos de metadados. Se o serviço encontrar vários pods com rótulos correspondentes, ele faz o balanceamento de carga das conexões de rede existentes entre eles.

Cada serviço recebe um endereço IP exclusivo para os clientes se conectarem a ele. Esse endereço IP vem de outro SDN interno do OpenShift, diferente da rede interna dos pods, mas visível somente para os pods. Cada pod correspondente ao selector é adicionado ao recurso do serviço como endpoint.

Descoberta de serviços

Um aplicativo normalmente encontra um endereço IP de serviço e uma porta usando variáveis de ambiente. Para cada serviço dentro de um projeto do OpenShift, as seguintes variáveis de ambiente são automaticamente definidas e injetadas em contêineres para todos os pods dentro do mesmo projeto:

- *SVC_NAME_SERVICE_HOST* é o endereço IP do serviço.
- *SVC_NAME_SERVICE_PORT* é a porta TCP do serviço.



nota

A parte *SVC_NAME* é alterada para cumprir com as restrições de nomenclatura DNS: as letras são maiúsculas e os sublinhados (_) são substituídos por traços (-).

Outra maneira de descobrir um serviço de um pod é usar o servidor DNS interno do OpenShift, que é visível somente para pods. A cada serviço, um registro SRV é dinamicamente atribuído com um FQDN do formulário:

```
SVC_NAME .PROJECT_NAME.svc.cluster.local
```

Ao descobrir serviços usando variáveis de ambiente, um pod deve ser criado e iniciado somente após o serviço ser criado. No entanto, se o aplicativo tiver sido feito para descobrir serviços usando consultas DNS, ele poderá encontrar serviços criados depois que o pod foi iniciado.

Para os aplicativos que precisam acessar o serviço de fora do cluster do OpenShift, existem duas maneiras de atingir este objetivo:

1. **Tipo NodePort:** esta é uma abordagem mais antiga baseada em Kubernetes, em que o serviço é disponibilizado a clientes externos, por meio das portas disponíveis no host do nó de computação que transmite as conexões ao endereço IP do serviço. Use o comando `oc edit svc` para editar os atributos do serviço, especifique `NodePort` como o valor para `type` e forneça um valor de porta para o atributo `nodePort`. O OpenShift, em seguida, transmite as conexões para o serviço por meio do endereço IP público do host do nó de computação e o valor da porta definido no `NodePort`.
2. **Rotas do OpenShift:** esta é a abordagem preferencial no OpenShift para expor serviços usando uma URL exclusiva. Use o comando `oc expose` para disponibilizar um serviço ao acesso externo ou disponibilizar um serviço do console da web do OpenShift.

Figura 6.5 ilustra como os serviços NodePort permitem acesso externo aos serviços do Kubernetes. As rotas do OpenShift são abordadas em mais detalhes neste curso.

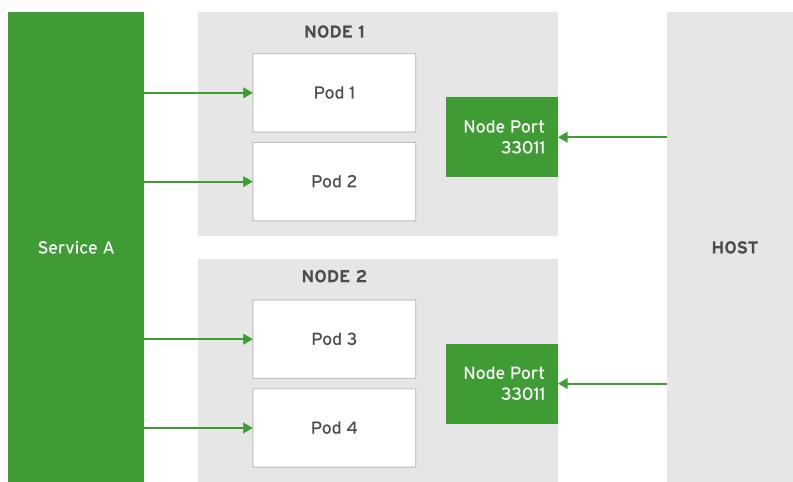


Figura 6.5: Método alternativo para acesso externo a um serviço do Kubernetes

O OpenShift oferece o comando `oc port-forward` para encaminhar uma porta local para uma porta de pod. Isso é diferente de ter acesso a um pod por meio de um recurso de serviço:

- O mapeamento de encaminhamento de porta existe somente na estação de trabalho na qual o cliente `oc` é executado, enquanto um serviço mapeia uma porta para todos os usuários da rede.
- Um serviço balanceia a carga de conexões para pods potencialmente múltiplos, enquanto um mapeamento de encaminhamento de porta encaminha conexões para um único pod.



nota

A Red Hat desencoraja o uso da abordagem `NodePort` para evitar expor o serviço a conexões diretas. Mapear via encaminhamento de porta no OpenShift é considerado uma alternativa mais segura.

O seguinte exemplo demonstra o uso do comando `oc port-forward`:

```
[user@host ~]$ oc port-forward mysql-openshift-1-glqrp 3306:3306
```

O comando encaminha a porta 3306 da máquina do desenvolvedor para a porta 3306 no pod `db`, onde um servidor MySQL (dentro de um contêiner) aceita conexões de rede.

**nota**

Ao executar esse comando, certifique-se de deixar a janela de terminal em execução. Fechar a janela ou cancelar o processo interrompe o mapeamento de porta.

Criação de aplicativos

Aplicativos simples, complexos de várias camadas e de microsserviço podem ser descritos por um único arquivo de definição de recursos. Esse arquivo conteria várias definições de pods, definições de serviços para conectar os pods, controladores de replicação ou Deployment para dimensionar horizontalmente os pods de aplicativos, PersistentVolumeClaims para persistir dados de aplicativos e tudo o que for necessário e que pode ser gerenciado pelo OpenShift.

O comando `oc new-app` pode ser usado com a opção `-o json` ou `-o yaml` para criar um arquivo de estrutura de definição de recurso no formato JSON ou YAML, respectivamente. Esse arquivo pode ser personalizado e usado para criar um aplicativo usando o comando `oc create -f <filename>` ou mesclado a outros arquivos de definição de recursos para criar aplicativos compostos.

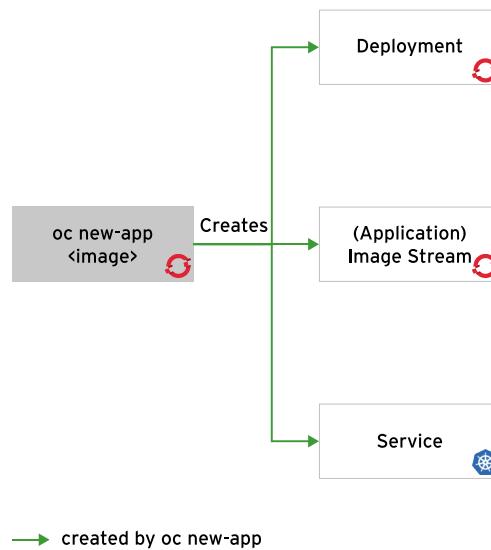
O comando `oc new-app` pode criar pods de aplicativos para execução no OpenShift de várias formas diferentes. Ele pode criar pods a partir de imagens existentes do Docker, a partir de arquivos do Docker e de código-fonte bruto usando o processo de Source-to-Image (S2I).

Execute o comando `oc new-app -h` para entender todas as diferentes opções disponíveis para criar novos aplicativos no OpenShift.

O seguinte comando cria um aplicativo baseado em uma imagem, `mysql`, do Docker Hub, com o rótulo definido como `db=mysql`:

```
[user@host ~]$ oc new-app mysql MYSQL_USER=user MYSQL_PASSWORD=pass
MYSQL_DATABASE=testdb -l db=mysql
```

A seguinte figura mostra os recursos do Kubernetes e do OpenShift criados pelo comando `oc new-app` quando o argumento é uma imagem de contêiner:

**Figura 6.6: Recursos criados para um novo aplicativo**

O seguinte comando cria um aplicativo com base em uma imagem de um registro privado de imagem do Docker:

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

O seguinte comando cria um aplicativo com base em código-fonte armazenado em um repositório Git:

```
oc new-app https://github.com/openshift/ruby-hello-world --name=ruby-hello
```

Você aprenderá mais sobre o processo Source-to-Image (S2I), os conceitos associados e maneiras mais avançadas de usar o `oc new-app` para compilar aplicativos para o OpenShift na próxima sessão.

O seguinte comando cria um aplicativo baseado em um template existente:

```
$ oc new-app \
> --template=mysql-persistent \
> -p MYSQL_USER=user1 -p MYSQL_PASSWORD=mypassword -p MYSQL_DATABASE=testdb \
> -p MYSQL_ROOT_PASSWORD=rootpassword -p VOLUME_CAPACITY=10Gi
...output omitted...
```

**nota**

Você aprenderá mais sobre templates no próximo capítulo.

Gerenciamento de armazenamento persistente

Além da especificação de imagens personalizadas, você pode criar armazenamento persistente e anexá-lo ao aplicativo. Dessa forma, você pode garantir que os dados não serão perdidos ao excluir os pods. Para listar objetos `PersistentVolume` em um cluster, use o comando `oc get pv`:

```
[admin@host ~]$ oc get pv
NAME      CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS      CLAIM     ...
pv0001    1Mi       RWO          Retain           Available   ...
pv0002    10Mi      RWX          Recycle          Available   ...
...output omitted...
```

Para ver a definição YAML para um determinado `PersistentVolume`, use o comando `oc get` com a opção `-o yaml`:

```
[admin@host ~]$ oc get pv pv0001 -o yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  creationTimestamp: ...value omitted...
  finalizers:
  - kubernetes.io/pv-protection
  labels:
    type: local
    name: pv0001
  resourceVersion: ...value omitted...
  selfLink: /api/v1/persistentvolumes/pv0001
  uid: ...value omitted...
spec:
  accessModes:
  - ReadWriteOnce
  capacity:
    storage: 1Mi
  hostPath:
    path: /data/pv0001
    type: ""
  persistentVolumeReclaimPolicy: Retain
status:
  phase: Available
```

Para adicionar mais objetos `PersistentVolume` a um cluster, use o comando `oc create`:

```
[admin@host ~]$ oc create -f pv1001.yaml
```



nota

O arquivo `pv1001.yaml` acima deve conter uma definição de volume persistente, semelhante em estrutura à saída do comando `oc get pv pv-name -o yaml`.

Solicitação de volumes persistentes

Quando um aplicativo exige armazenamento, você cria um objeto `PersistentVolumeClaim` (PVC) para solicitar um recurso de armazenamento dedicado do pool de clusters. O conteúdo a seguir de um arquivo chamado `pvc.yaml` é uma definição para um PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myapp
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

O PVC define os requisitos de armazenamento para o aplicativo, como capacidade ou taxa de transferência. Para criar o PVC, use o comando `oc create`:

```
[admin@host ~]$ oc create -f pvc.yaml
```

Depois de criar um PVC, o OpenShift tenta encontrar um recurso `PersistentVolume` que satisfaça os requisitos do PVC. Se o OpenShift encontrar uma correspondência, ele vinculará o objeto `PersistentVolume` ao objeto `PersistentVolumeClaim`. Para listar os PVCs em um projeto, use o comando `oc get pvc`:

```
[admin@host ~]$ oc get pvc
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
myapp    Bound    pv0001   1Gi        RWO          
```

A saída indica se um volume persistente está vinculado ao PVC, junto com os atributos do PVC (como a capacity).

Para usar o volume persistente em um pod do aplicativo, defina uma montagem de volume para um contêiner que faça referência ao objeto `PersistentVolumeClaim`. A definição do pod de aplicativo abaixo faz referência ao objeto `PersistentVolumeClaim` para definir uma montagem de volume para o aplicativo:

```
apiVersion: "v1"
kind: "Pod"
metadata:
  name: "myapp"
  labels:
    name: "myapp"
spec:
  containers:
    - name: "myapp"
      image: openshift/myapp
      ports:
        - containerPort: 80
          name: "http-server"
  volumeMounts:
```

```

    - mountPath: "/var/www/html"
      name: "pvol" ①
  volumes:
    - name: "pvol" ②
      persistentVolumeClaim:
        claimName: "myapp" ③

```

- ① Esta seção declara que o volume `pvol` é montado em `/var/www/html` no sistema de arquivos do contêiner.
- ② Esta seção define o volume `pvol`.
- ③ O volume `pvol` faz referência ao PVC `myapp`. Se o OpenShift associa um volume persistente disponível ao PVC `myapp`, o volume `pvol` refere-se a esse volume associado.

Gerenciamento de recursos do OpenShift na linha de comando

Há vários comandos essenciais usados para gerenciar recursos do OpenShift, como descrito abaixo.

Use o comando `oc get` para recuperar informações sobre recursos no cluster. Normalmente, esse comando mostra apenas as características mais importantes dos recursos e omite as informações mais detalhadas.

O comando `oc get RESOURCE_TYPE` exibe um resumo de todos os recursos do tipo especificado. O exemplo a seguir ilustra a saída de exemplo do comando `oc get pods`.

NAME	READY	STATUS	RESTARTS	AGE
<code>nginx-1-5r583</code>	1/1	Running	0	1h
<code>myapp-1-l44m7</code>	1/1	Running	0	1h

oc get all

Use o comando `oc get all` para recuperar um resumo dos componentes mais importantes de um cluster. Esse comando faz a iteração por meio dos principais tipos de recursos para o projeto atual e imprime um sumário das suas informações:

NAME	DOCKER REPO	TAGS	UPDATED	
<code>is/nginx</code>	<code>172.30.1.1:5000/basic-kubernetes/nginx</code>	<code>latest</code>	About an hour ago	
NAME	REVISION	DESIRED	CURRENT	TRIGGERED BY
<code>dc/nginx</code>	1	1	1	<code>config,image(nginx:latest)</code>
NAME	DESIRED	CURRENT	READY	AGE
<code>rc/nginx-1</code>	1	1	1	1h
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
<code>svc/nginx</code>	<code>172.30.72.75</code>	<code><none></code>	<code>80/TCP, 443/TCP</code>	1h
NAME	READY	STATUS	RESTARTS	AGE
<code>po/nginx-1-ypp8t</code>	1/1	Running	0	1h

oc describe

Se os sumários fornecidos pelo `oc get` forem insuficientes, use o comando `oc describe RESOURCE_TYPE RESOURCE_NAME` para recuperar as informações adicionais. Diferente do comando `oc get`, não há maneira de iterar todos os diferentes recursos por tipo. Embora a maioria dos principais recursos possa ser descrita, essa funcionalidade não está disponível em todos os recursos. Esta é uma amostra de saída da descrição de um recurso de pod:

```
Name: mysql-openshift-1-glqrp
Namespace: mysql-openshift
Priority: 0
Node: cluster-worker-1/172.25.250.52
Start Time: Fri, 15 Feb 2019 02:14:34 +0000
Labels: app=mysql-openshift
        deployment=mysql-openshift-1
...output omitted...
Status: Running
IP: 10.129.0.85
```

oc get

O comando `oc get RESOURCE_TYPE RESOURCE_NAME` pode ser usado para exportar uma definição de recurso. Casos típicos de uso abrangem a criação de um backup ou o auxílio na modificação de uma definição. A opção `-o yaml` imprime a representação do objeto em formato YAML, mas isso pode ser modificado para o formato JSON, fornecendo uma opção `-o json`.

oc create

Esse comando cria recursos a partir da definição de um recurso. Normalmente, isso é emparelhado com o comando `oc get RESOURCE_TYPE RESOURCE_NAME -o yaml` para editar definições.

oc edit

Esse comando permite ao usuário editar recursos de uma definição de um recurso. Por padrão, esse comando abre um buffer `vi` para editar a definição do recurso.

oc delete

O comando `oc delete RESOURCE_TYPE name` remove um recurso de um cluster do OpenShift. Observe que uma compreensão fundamental da arquitetura do OpenShift é necessária aqui, pois a exclusão de recursos gerenciados como pods resulta em novas instâncias daqueles recursos que estão sendo criados automaticamente. Quando um projeto é excluído, ele exclui todos os recursos e aplicativos contidos nele.

oc exec

O comando `oc exec CONTAINER_ID options` executa comandos dentro de um contêiner. Você pode usar esse comando para executar comandos de lotes interativos e não interativos como parte do script.

Recursos de rotulagem

Ao trabalhar com muitos recursos no mesmo projeto, geralmente é útil agrupar esses recursos por aplicativo, ambiente ou outros critérios. Para estabelecer esses grupos, você define rótulos

para os recursos em seu projeto. Rótulos fazem parte da seção de `metadata` de um recurso e são definidas como pares de chave–valor, conforme mostrado no exemplo a seguir:

```
apiVersion: v1
kind: Service
metadata:
...contents omitted...
labels: app: nexus template: nexus-persistent-template
name: nexus
...contents omitted...
```

Muitos subcomandos `oc` são compatíveis com uma opção `-l` para processar recursos a partir de uma especificação de rótulo. Para o comando `oc get`, a opção `-l` atua como um seletor para recuperar apenas objetos que tenham um rótulo correspondente:

```
$ oc get svc,deployments -l app=nexus
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
service/nexus  ClusterIP  172.30.29.218  <none>        8081/TCP    4h

NAME           REVISION      DESIRED      CURRENT      ...
deployment.apps.openshift.io/nexus  1           1           1           ...
```

**nota**

Embora qualquer rótulo possa ser usado em recursos, as chaves `app` e `template` são comuns para rótulos. Por convenção, a chave `app` indica o aplicativo relacionado a esse recurso. A chave `template` rotula todos os recursos gerados pelo mesmo modelo com o nome do modelo.

Ao usar modelos para gerar recursos, os rótulos são especialmente úteis. Um recurso de template tem uma seção de `labels` separada da seção `metadata.labels`. Os rótulos definidos na seção `labels` não se aplicam ao modelo em si, mas são adicionados a todos os recursos gerados pelo template:

```
apiVersion: template.openshift.io/v1
kind: Template
labels: app: nexus template: nexus-persistent-template
metadata:
...contents omitted...
labels: maintainer: redhat
  name: nexus-persistent
...contents omitted...
objects:
- apiVersion: v1
  kind: Service
  metadata:
    name: nexus
labels: version: 1
...contents omitted...
```

O exemplo anterior define um recurso de template com um único rótulo: `maintainer: redhat`. O modelo gera um recurso de serviço com três rótulos: `app: nexus`, `template: nexus-persistent-template` e `version: 1`.



Referências

Informações adicionais sobre pods e serviços podem ser encontradas na seção *Pods and Services* do documento do OpenShift Container Platform:

Arquitetura

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/architecture/index
>

Informações adicionais sobre a criação de imagens estão disponíveis na documentação do OpenShift Container Platform:

Criação de imagens

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/images/index
>

Os rótulos e os seletores de rótulos estão disponíveis na seção *Working with Kubernetes Objects* da documentação do Kubernetes:

Rótulos e seletores

<https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/>
>

► Exercício Guiado

Implantação de um servidor de banco de dados no OpenShift

Neste exercício, você criará e implantará um pod de banco de dados do MySQL usando o comando `oc new-app`.

Resultados

Você deverá ser capaz de criar e implantar um pod de banco de dados MySQL no OpenShift.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Na workstation, execute o seguinte comando para configurar o ambiente:

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

Instruções

- 1. Prepare o ambiente de laboratório.

- 1.1. Carregue a configuração do ambiente de sala de aula.

Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- 1.2. Faça login no cluster do OpenShift.

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

- 1.3. Crie um novo projeto que contenha o nome de usuário do desenvolvedor RHOCUP para os recursos que você criar durante este exercício:

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

- 2. Crie um novo aplicativo a partir do template `mysql-persistent` usando o comando `oc new-app`.

A imagem exige que você use a opção `-p` para definir as variáveis de ambiente `MYSQL_USER`, `MYSQL_PASSWORD`, `MYSQL_DATABASE`, `MYSQL_ROOT_PASSWORD` e `VOLUME_CAPACITY`.

Use a opção `--template` com o comando `oc new-app` para especificar um template com armazenamento persistente para que o OpenShift não faça pull da imagem da Internet:

```
[student@workstation ~]$ oc new-app \
> --template=mysql-persistent \
> -p MYSQL_USER=user1 -p MYSQL_PASSWORD=mypa55 -p MYSQL_DATABASE=testdb \
> -p MYSQL_ROOT_PASSWORD=r00tpa55 -p VOLUME_CAPACITY=10Gi
--> Deploying template "openshift/mysql-persistent" to project
${RHT_OCP4_DEV_USER}-mysql-openshift
...output omitted...
--> Creating resources ...
secret "mysql" created
service "mysql" created
persistentvolumeclaim "mysql" created
deploymentconfig.apps.openshift.io "mysql" created
--> Success
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose service/mysql'
Run 'oc status' to view your app.
```

- ▶ 3. Verifique se o pod do MySQL foi criado com êxito e visualize os detalhes sobre o pod e seu serviço.
 - 3.1. Execute o comando `oc status` para visualizar o status do novo aplicativo e para verificar se a implantação da imagem do MySQL teve êxito:

```
[student@workstation ~]$ oc status
In project ${RHT_OCP4_DEV_USER}-mysql-openshift on server ...
svc/mysql - 172.30.151.91:3306
...output omitted...
deployment #1 deployed 6 minutes ago - 1 pod
```

- 3.2. Liste os pods neste projeto para verificar se o pod do MySQL está pronto e em execução:

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
mysql-1-5vfn4  1/1     Running   0          109s
```



nota

Observe o nome do pod em execução. Você precisará dessa informação para fazer login no servidor de banco de dados MySQL posteriormente.

- 3.3. Use o comando `oc describe` para visualizar mais detalhes sobre o pod:

```
[student@workstation ~]$ oc describe pod mysql-1-5vfn4
Name:           mysql-1-5vfn4
Namespace:      ${RHT_OCP4_DEV_USER}-mysql-openshift
Priority:       0
Node:           master01/192.168.50.10
Start Time:     Mon, 29 Mar 2021 16:42:13 -0400
Labels:         deployment=mysql-1
...output omitted...
Status:         Running
IP:             10.10.0.34
...output omitted...
```

- 3.4. Liste os serviços neste projeto e verifique se o serviço de acesso ao MySQL foi criado:

```
[student@workstation ~]$ oc get svc
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
mysql         ClusterIP  172.30.151.91  <none>        3306/TCP    10m
```

- 3.5. Recupere os detalhes do serviço mysql usando o comando `oc describe` e observe se o tipo de serviço é `ClusterIP` por padrão:

```
[student@workstation ~]$ oc describe service mysql
Name:           mysql
Namespace:      ${RHT_OCP4_DEV_USER}-mysql-openshift
Labels:         app=mysql-persistent
                app.kubernetes.io/component=mysql-persistent
                app.kubernetes.io/instance=mysql-persistent
                template=mysql-persistent-template
Annotations:    openshift.io/generated-by: OpenShiftNewApp
...output omitted...
Selector:       name=mysql
Type:           ClusterIP
IP Family Policy: SingleStack
IP Families:   IPv4
IP:             172.30.151.91
IPs:            172.30.151.91
Port:           mysql  3306/TCP
TargetPort:     3306/TCP
Endpoints:     10.10.0.34:3306
Session Affinity: None
Events:         <none>
```

- 3.6. Liste as afirmações de armazenamento persistente neste projeto:

```
[student@workstation ~]$ oc get pvc
NAME      STATUS    VOLUME                                     CAPACITY  ...
STORAGECLASS
mysql     Bound     pvc-e9bf0b1f-47df-4500-afb6-77e826f76c15  10Gi     ...
standard
```

- 3.7. Recupere os detalhes da PVC mysql usando o comando `oc describe`:

```
[student@workstation ~]$ oc describe pvc/mysql
Name: mysql
Namespace: ${RHT_OCP4_DEV_USER}-mysql-openshift
StorageClass: standard
Status: Bound
Volume: pvc-e9bf0b1f-47df-4500-afb6-77e826f76c15
Labels: app=mysql-persistent
        app.kubernetes.io/component=mysql-persistent
        app.kubernetes.io/instance=mysql-persistent
        template=mysql-persistent-template
Annotations: openshift.io/generated-by: OpenShiftNewApp
...output omitted...
Capacity: 10Gi
Access Modes: RWO
VolumeMode: Filesystem
Used By: mysql-1-5vfn4
...output omitted...
```

- 4. Conecte-se ao servidor do banco de dados do MySQL e verifique se o banco de dados foi criado com êxito.

- 4.1. Na máquina `workstation`, configure o encaminhamento de porta entre a `workstation` e o pod de banco de dados em execução no OpenShift usando a porta 3306. O terminal travará depois de executar o comando.

```
[student@workstation ~]$ oc port-forward mysql-1-5vfn4 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
```

- 4.2. Na máquina `workstation`, abra outro terminal e conecte-se ao servidor MySQL usando o cliente MySQL.

```
[student@workstation ~]$ mysql -uuser1 -pmypa55 --protocol tcp -h localhost
mysql: [Warning] Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 13
Server version: 8.0.26 Source distribution

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective

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

mysql>
```

- 4.3. Verifique a criação do banco de dados `testdb`.

```
mysql> show databases;  
-----  
| Database      |  
-----  
| information_schema |  
| testdb          |  
-----  
2 rows in set (0.00 sec)
```

4.4. Saia do prompt do MySQL:

```
mysql> exit  
Bye
```

Feche o terminal e retorne ao anterior. Conclua o processo de encaminhamento de porta pressionando **Ctrl+C**.

```
Forwarding from 127.0.0.1:3306 -> 3306  
Forwarding from [::1]:3306 -> 3306  
Handling connection for 3306  
^C
```

► 5. Exclua o projeto para remover todos os recursos no projeto:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-mysql-openshift
```

Encerramento

Na workstation, execute o script `lab openshift-resources finish` para concluir esse laboratório.

```
[student@workstation ~]$ lab openshift-resources finish
```

Isso conclui o exercício.

Criação de rotas

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de expor serviços usando rotas do OpenShift.

Trabalho com rotas

Os serviços permitem acesso à rede entre os pods dentro de uma instância do OpenShift e as rotas dão permissão de acesso à rede aos pods de fora da instância do OpenShift.

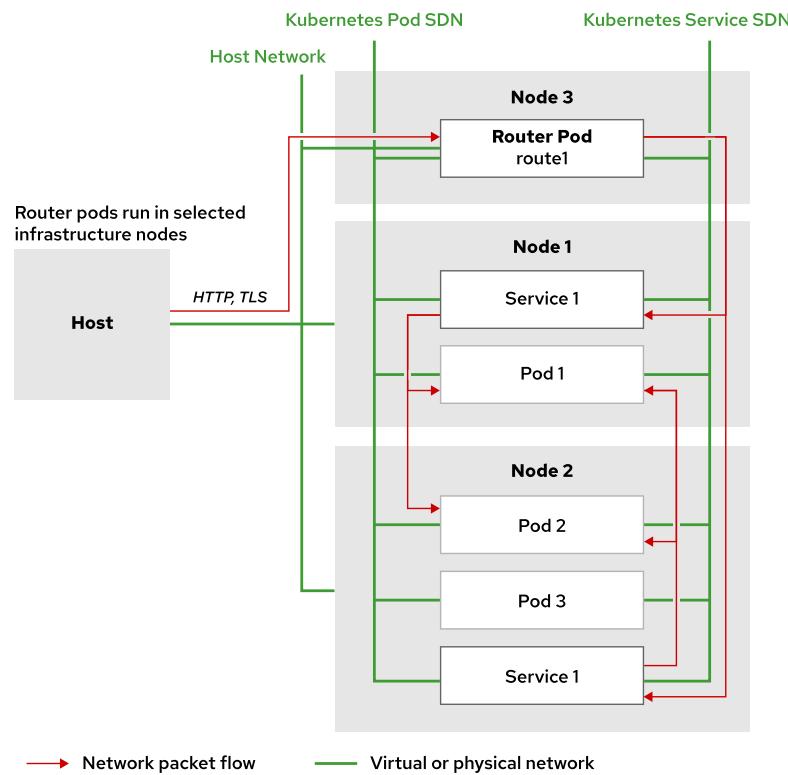


Figura 6.7: As rotas do OpenShift e os serviços do Kubernetes

Uma rota conecta um endereço IP voltado ao público externo e um nome de host DNS a um IP de serviço voltado ao público interno. Ele usa o recurso do serviço para encontrar os pontos de extremidade, ou seja, as portas expostas pelo serviço.

As rotas do OpenShift são implementadas por um serviço de roteador em todo o cluster, que é executado como um aplicativo conteinerizado no cluster do OpenShift. O OpenShift escala e replica pods de roteadores como qualquer outro aplicativo OpenShift.

**nota**

Na prática, para melhorar o desempenho e reduzir a latência, o roteador do OpenShift se conecta diretamente aos pods por meio da rede interna definida por software do pod (SDN).

O serviço de roteador usa o *HAProxy* como a implementação padrão.

Uma consideração importante para os administradores do OpenShift é que os nomes de host DNS públicos, configurados para as rotas, precisam apontar para os endereços IP públicos dos nós que executam o roteador. Os pods de roteadores, ao contrário dos pods de aplicativos convencionais, se associam aos endereços IP públicos de seus nós, em vez de à SDN interna do pod.

O seguinte exemplo mostra uma rota mínima definida por meio da sintaxe JSON:

```
{
  "apiVersion": "v1",
  "kind": "Route",
  "metadata": {
    "name": "quoteapp"
  },
  "spec": {
    "host": "quoteapp.apps.example.com",
    "to": {
      "kind": "Service",
      "name": "quoteapp"
    }
  }
}
```

Os atributos `apiVersion`, `kind` e `metadata` seguem as regras de definição de recursos padrão do Kubernetes. O valor `Route` para `kind` mostra se isso é um recurso de rota, e o atributo `metadata.name` dá o identificador `quoteapp` a essa rota específica.

Assim como com pods e serviços, a parte principal é o atributo `spec`, que é um objeto contendo os seguintes atributos:

- `host` é uma string contendo o FQDN associado à rota. O DNS deve resolver esse FQDN para o endereço IP do roteador do OpenShift. Os detalhes para modificar a configuração do DNS estão fora do escopo deste curso.
- `to` é um objeto que indica o recurso para o qual essa rota direciona. Nesse caso, a rota direciona para um serviço do OpenShift com o `name` definido como `quoteapp`.

**nota**

Nomes de tipos de recurso diferentes não entram em conflito. É perfeitamente aceitável ter uma rota de nome `quoteapp` que aponta um serviço também denominado `quoteapp`.

**Importante**

Ao contrário dos serviços, que usam seletores para se vincularem a recursos do pod contendo rótulos específicos, uma rota é vinculada diretamente ao nome do recurso do serviço.

Criação de rotas

Use o comando `oc create` para criar recursos de rota, como qualquer outro recurso do OpenShift. Você deve fornecer um arquivo de definição de recurso JSON ou YAML, que defina a rota, para o comando `oc create`.

O comando `oc new-app` não cria um recurso de rota na hora de compilar um pod a partir de imagens de contêiner, Containerfiles ou código-fonte do aplicativo. Afinal, o `oc new-app` não sabe se o pod deve ficar acessível do lado de fora da instância do OpenShift ou não.

Outra maneira de criar uma rota é por meio do comando `oc expose service`, passando o nome do recurso do serviço como entrada. A opção `--name` pode ser usada para controlar o nome do recurso da rota. Por exemplo:

```
$ oc expose service quotedb --name quote
```

Por padrão, as rotas criadas por `oc expose` geram nomes DNS do formulário: `route-name-project-name.default-domain`

Em que:

- `route-name` é o nome atribuído à rota. Se nenhum nome explícito for definido, o OpenShift atribui à rota o mesmo nome do recurso de origem (como o nome do serviço).
- `project-name` é o nome do projeto que possui o recurso.
- O `default-domain` está configurado no plano de controle do OpenShift e estabelece que um domínio DNS curinga é um pré-requisito para a instalação do OpenShift.

Por exemplo, a criação de uma rota chamada `quote` no projeto chamado `test` a partir da instância do OpenShift em que o domínio curinga é `cloudapps.example.com` resulta no FQDN `quote-test.cloudapps.example.com`.

**nota**

O servidor DNS que hospeda o domínio curinga não sabe nada sobre os nomes do host da rota. Ele simplesmente resolve qualquer nome para os endereços IP configurados. Somente o roteador OpenShift reconhece os nomes de host das rotas, tratando cada um como um host virtual HTTP. O roteador do OpenShift bloqueia nomes de host de domínio curinga inválidos que não correspondem a nenhuma rota, gerando um erro HTTP 404.

Aproveitamento do serviço de roteamento padrão

O serviço de roteamento padrão é implementado como um pod do HAProxy. Os pods de roteador, os contêineres e suas configurações podem ser inspecionados como qualquer outro recurso em um cluster do OpenShift:

```
$ oc get pod --all-namespaces | grep router
openshift-ingress  router-default-746b5cfb65-f6sdm 1/1     Running  1          4d
```

Você pode consultar informações sobre o roteador padrão usando o rótulo associado, como mostrado aqui.

Por padrão, o roteador é implantado no projeto `openshift-ingress`. Use o comando `oc describe pod` para obter os detalhes de configuração de roteamento:

```
$ oc describe pod router-default-746b5cfb65-f6sdm
Name:           router-default-746b5cfb65-f6sdm
Namespace:      openshift-ingress
...output omitted...
Containers:
  router:
    ...output omitted...
    Environment:
      STATS_PORT:          1936
      ROUTER_SERVICE_NAMESPACE:  openshift-ingress
      DEFAULT_CERTIFICATE_DIR: /etc/pki/tls/private
      ROUTER_SERVICE_NAME:    default
      ROUTER_CANONICAL_HOSTNAME: apps.cluster.lab.example.com
...output omitted...
```

O subdomínio, ou domínio padrão a ser usado em todas as rotas padrão, recebe seu valor da entrada `ROUTER_CANONICAL_HOSTNAME`.



Referências

Informações adicionais sobre a arquitetura das rotas no OpenShift estão disponíveis nas seções *Architecture* e *Developer Guide* do **Documentação do OpenShift Container Platform**.

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/

► Exercício Guiado

Exposição de um serviço como uma rota

Neste exercício, você criará, compilará e implantará um aplicativo em um cluster do OpenShift e exporá seu serviço como uma rota.

Resultados

Você deverá ser capaz de expor um serviço como uma rota para um aplicativo do OpenShift implantado.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Abra um terminal na **workstation** como usuário **student** e execute o seguinte comando:

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

Instruções

- 1. Prepare o ambiente de laboratório.

- 1.1. Carregue a configuração do ambiente de sala de aula.

Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- 1.2. Faça login no cluster do OpenShift.

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

- 1.3. Crie um novo projeto que contenha o nome de usuário do desenvolvedor RHOCP para os recursos que você criar durante este exercício.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-route
```

- 2. Crie um novo aplicativo PHP usando a imagem `quay.io/redhattraining/php-hello-dockerfile`.

- 2.1. Use o comando `oc new-app` para criar um aplicativo PHP.

**Importante**

O exemplo a seguir usa uma barra invertida (\) para indicar que a segunda linha é uma continuação da primeira. Se você deseja ignorar a barra invertida, é possível digitar todo o comando na primeira linha.

```
[student@workstation ~]$ oc new-app \
> --docker-image=quay.io/redhattraining/php-hello-dockerfile \
> --name php-helloworld
--> Found container image 4b696cc (2 years old) from quay.io for "quay.io/
redhattraining/php-hello-dockerfile"
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose service/php-helloworld'
Run 'oc status' to view your app.
```

- 2.2. Aguarde até que o aplicativo conclua a implantação monitorando o progresso com o comando `oc get pods -w`:

```
[student@workstation ~]$ oc get pods -w
NAME                      READY   STATUS    RESTARTS   AGE
php-helloworld-74bb86f6cb-zt6wl   1/1     Running   0          5s
^C
```

Sua saída exata pode diferir em nomes, status, tempo e ordem. O contêiner no status `Running` com um sufixo aleatório (`74bb86f6cb-zt6wl` no exemplo) contém o aplicativo e mostra que ele está ativo e em execução. Como alternativa, monitore os logs de implantação com o comando `oc logs -f php-helloworld-74bb86f6cb-zt6wl`. Pressione Ctrl + C para sair do comando se necessário.

```
[student@workstation ~]$ oc logs -f php-helloworld-74bb86f6cb-zt6wl
AH00558: httpd: Could not reliably determine the server's fully qualified domain
name, using 10.129.5.124. Set the 'ServerName' directive globally to suppress
this message
[09-Aug-2021 22:09:45] NOTICE: [pool www] 'user' directive is ignored when FPM is
not running as root
[09-Aug-2021 22:09:45] NOTICE: [pool www] 'group' directive is ignored when FPM is
not running as root
`^C`
```

Sua saída exata pode ser diferente.

- 2.3. Revise o serviço para esse aplicativo usando o comando `oc describe`:

```
[student@workstation ~]$ oc describe svc/php-helloworld
Name:          php-helloworld
Namespace:     extrdp-route
Labels:        app=php-helloworld
               app.kubernetes.io/component=php-helloworld
               app.kubernetes.io/instance=php-helloworld
Annotations:   openshift.io/generated-by: OpenShiftNewApp
Selector:      deployment=php-helloworld
Type:          ClusterIP
IP:            172.30.100.236
Port:          8080-tcp  8080/TCP
TargetPort:    8080/TCP
Endpoints:    10.129.5.124:8080
Session Affinity: None
Events:        <none>
```

O endereço IP e o namespace exibidos na saída do comando podem ser diferentes.

- ▶ 3. Exponha o serviço; isso cria uma rota: Use o nome padrão e o nome de domínio totalmente qualificado (FQDN) para a rota:

```
[student@workstation ~]$ oc expose svc/php-helloworld
route.route.openshift.io/php-helloworld exposed
[student@workstation ~]$ oc describe route
Name:          php-helloworld
Namespace:     extrdp-route
Created:       16 seconds ago
Labels:        app=php-helloworld
               app.kubernetes.io/component=php-helloworld
               app.kubernetes.io/instance=php-helloworld
Annotations:   openshift.io/host.generated=true
Requested Host: php-helloworld-extrdp-route.apps.na46-stage2.dev.nextcle.com
               exposed on router default (host apps.na46-
stage2.dev.nextcle.com) 16 seconds ago
Path:          <none>
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port:  8080-tcp

Service:      php-helloworld
Weight:       100 (100%)
Endpoints:   10.129.5.124:8080
```

- 4. Acesse o serviço a partir de um host externo ao cluster para verificar se o serviço e a rota estão funcionando.

```
[student@workstation ~]$ curl \
> php-helloworld-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! PHP version is 7.2.11
```

**nota**

A saída do aplicativo PHP depende da versão real da imagem. Ela pode ser diferente da sua.

O FQDN é, por padrão, composto pelo nome do aplicativo e pelo nome do projeto. O restante do FQDN, o subdomínio, é definido quando o OpenShift é instalado.

- 5. Substitua essa rota por uma rota chamada xyz.

5.1. Exclua a rota atual:

```
[student@workstation ~]$ oc delete route/php-helloworld
route.route.openshift.io "php-helloworld" deleted
```

**nota**

A exclusão da rota é opcional. Você pode ter várias rotas para o mesmo serviço, desde que tenham nomes diferentes.

5.2. Crie uma rota para o serviço com um nome de \${RHT_OCP4_DEV_USER} - xyz.

```
[student@workstation ~]$ oc expose svc/php-helloworld \
> --name=${RHT_OCP4_DEV_USER}-xyz
route.route.openshift.io/${RHT_OCP4_DEV_USER}-xyz exposed
[student@workstation ~]$ oc describe route
Name:           extrdp-xyz
Namespace:      extrdp-route
Created:        23 seconds ago
Labels:         app=php-helloworld
                app.kubernetes.io/component=php-helloworld
                app.kubernetes.io/instance=php-helloworld
Annotations:   openshift.io/host.generated=true
Requested Host: extrdp-xyz-extrdp-route.apps.na46-stage2.dev.nextcle.com
                exposed on router default (host apps.na46-
stage2.dev.nextcle.com) 22 seconds ago
Path:           <none>
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port:  8080-tcp

Service:     php-helloworld
Weight:      100 (100%)
Endpoints:   10.129.5.124:8080
```

Sua saída exata pode ser diferente. Observe o novo FQDN que foi gerado com base no novo nome da rota. Tanto o nome da rota quanto o nome do projeto contêm seu nome de usuário, por isso ele aparece duas vezes no FQDN da rota.

- 5.3. Faça uma solicitação HTTP usando o FQDN na porta 80:

```
[student@workstation ~]$ curl \
> ${RHT_OCP4_DEV_USER}-xyz-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! PHP version is 7.2.11
```

Encerramento

Na workstation, execute o script `lab openshift-routes finish` para concluir esse exercício.

```
[student@workstation ~]$ lab openshift-routes finish
```

Isso conclui o exercício orientado.

Criação de aplicativos com Source-to-Image

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de implantar um aplicativo usando o recurso Source-to-Image (S2I) do OpenShift Container Platform.

O processo Source-to-Image (S2I)

Source-to-Image (S2I) é uma ferramenta que facilita a compilação de imagens de contêiner a partir de um código-fonte de aplicativo. Essa ferramenta obtém o código-fonte de um aplicativo no repositório Git, injeta o código-fonte e um contêiner de base de acordo com a linguagem e estrutura desejadas e produz uma nova imagem de contêiner que executa o aplicativo montado.

Esta figura mostra os recursos criados pelo comando `oc new-app` quando o argumento é um repositório de código-fonte de aplicativo. Observe que o S2I também cria uma implantação e todos os seus recursos dependentes:

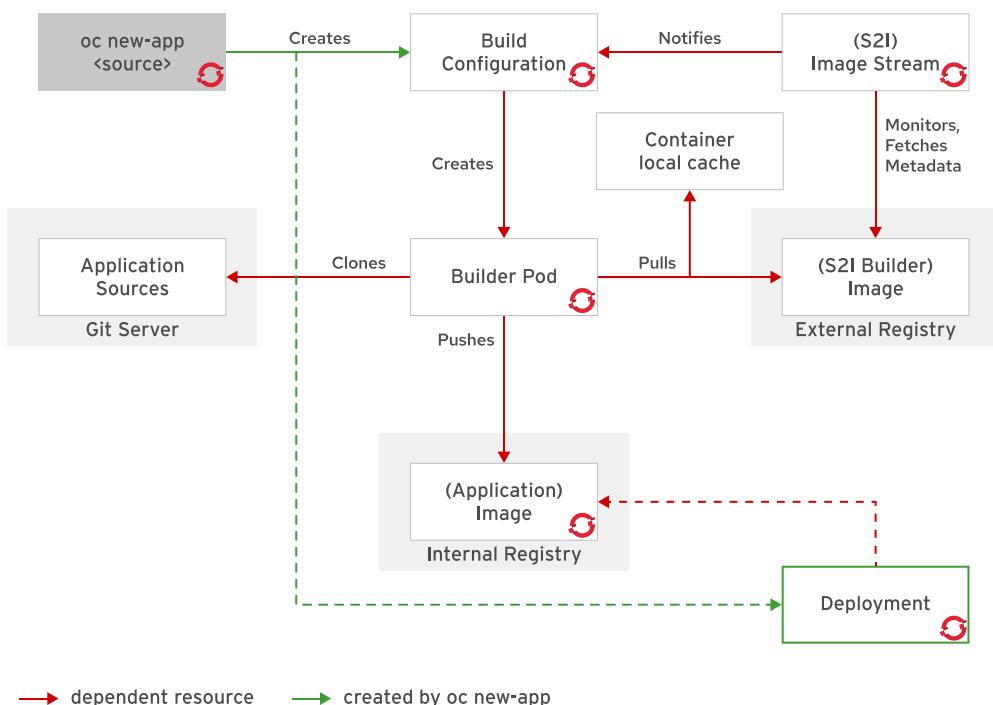


Figura 6.8: Implantação e recursos dependentes

O S2I é a principal estratégia usada para compilar aplicativos no OpenShift Container Platform. As principais razões de se usar compilações de fonte são:

- **Eficiência para o usuário:** os desenvolvedores não precisam entender os arquivos do Docker e os comandos do sistema operacional, como o yum install. Eles trabalham usando as ferramentas padrão de linguagens de programação.

- Aplicação de patch: o S2I permite a recriação de todos os aplicativos de forma consistente, se uma imagem de base precisar de um patch devido a um problema de segurança. Por exemplo, se um problema de segurança for encontrado em uma imagem de base PHP, a atualização dessa imagem com patches de segurança atualiza todos os aplicativos que a usam como base.
- Velocidade: com o S2I, o processo de montagem pode realizar um grande número de operações complexas sem criar uma nova camada em cada etapa, resultando em compilações mais rápidas.
- Ecossistema: o S2I encoraja um ecossistema compartilhado de imagens em que imagens de base e scripts podem ser personalizados e reutilizados em vários tipos de aplicativos.

Descrição de fluxos de imagem

O OpenShift implanta rapidamente novas versões de aplicativos de usuário nos pods. Para criar um novo aplicativo, além do código-fonte do aplicativo, uma imagem de base (a imagem de base S2I) é necessária. Se um desses dois componentes for atualizado, o OpenShift cria uma nova imagem de contêiner. Pods criados usando a imagem de contêiner antiga são substituídos por pods criados usando a nova imagem.

Embora seja evidente que a imagem de contêiner precisa ser atualizada quando o código de aplicativo for alterado, pode não ser evidente que os pods implantados também precisam ser atualizados se a imagem do construtor é alterada.

O recurso de fluxo de imagem é uma configuração que denomina imagens de contêiner específicas associadas a tags de fluxo de imagem, um alias para essas imagens de contêiner. O OpenShift compila aplicativos com base em um fluxo de imagem. O instalador do OpenShift preenche vários fluxos de imagem por padrão durante a instalação. Para determinar os fluxos de imagem disponíveis, use o comando `oc get`, conforme a seguir:

```
$ oc get is -n openshift
NAME      IMAGE REPOSITORY          TAGS
cli       ...svc:5000/openshift/cli    latest
dotnet    ...svc:5000/openshift/dotnet   2.1,...,3.1-el7,latest
dotnet-runtime ...svc:5000/openshift/dotnet-runtime 2.1,...,3.1-el7,latest
httpd     ...svc:5000/openshift/httpd    2.4,2.4-el7,2.4-el8,latest
jenkins   ...svc:5000/openshift/jenkins  2,latest
mariadb   ...svc:5000/openshift/mariadb  10.3,10.3-el7,10.3-el8,latest
mongodb   ...svc:5000/openshift/mongodb  3.6,latest
mysql     ...svc:5000/openshift/mysql    8.0,8.0-el7,8.0-el8,latest
nginx     ...svc:5000/openshift/nginx    1.10,1.12,1.8,latest
nodejs    ...svc:5000/openshift/nodejs   10,...,12-ubi7,12-ubi8
perl      ...svc:5000/openshift/perl    5.26,...,5.30,5.30-el7
php       ...svc:5000/openshift/php     7.2-ubi8,...,7.3-ubi8,latest
postgresql ...svc:5000/openshift/postgresql 10,10-el7,...,12-el7,12-el8
python    ...svc:5000/openshift/python  2.7,2.7-ubi7,...,3.6-ubi8,3.8
redis     ...svc:5000/openshift/redis   5,5-el7,5-el8,latest
ruby      ...svc:5000/openshift/ruby    2.5,2.5-ubi7,...,2.6,2.6-ubi7
...
```



nota

Sua instância do OpenShift pode ter mais ou menos fluxos de imagem, dependendo das adições locais e das versões do ponto do OpenShift.

O OpenShift detecta quando um fluxo de imagem muda e de tomar providências com base em tal mudança. Se surgir um problema de segurança na imagem `rhel8/nodejs-10`, ele pode ser atualizado no repositório de imagem e o OpenShift pode acionar automaticamente uma nova compilação do código do aplicativo.

É provável que uma organização escolha várias imagens S2I de base compatíveis da Red Hat, mas também pode criar suas próprias imagens de base.

Compilação de um aplicativo com S2I e a CLI

A compilação de um aplicativo com S2I pode ser realizada através da CLI do OpenShift.

Um aplicativo pode ser criado usando-se o processo S2I com o comando `oc new-app` da CLI:

```
$ oc new-app php~http://my.git.server.com/my-app ①②  
--name=myapp ③
```

- ① O fluxo de imagem usado no processo aparece à esquerda do caractere til (~).
- ② A URL depois do til indica a localização do repositório Git do código-fonte.
- ③ Define o nome do aplicativo.



nota

Em vez de usar o til, você pode definir o fluxo de imagem usando a opção `-i` ou `--image-stream` para obter a versão completa.

```
$ oc new-app -i php http://services.lab.example.com/app --name=myapp
```

O comando `oc new-app` permite a criação de aplicativos usando o código-fonte de um repositório Git local ou remoto. Se somente um repositório de fonte for especificado, `oc new-app` tentará identificar o fluxo de imagem correto a ser usado para compilar o aplicativo. Além do código do aplicativo, o S2I também pode identificar e processar arquivos do Docker para criar uma nova imagem.

O exemplo a seguir cria um aplicativo usando o repositório Git no diretório atual:

```
$ oc new-app .
```



Importante

Ao usar um repositório Git local, o repositório deve ter uma origem remota que aponta para uma URL acessível pela instância do OpenShift.

Também é possível criar um aplicativo usando um repositório Git remoto e um subdiretório de contexto:

```
$ oc new-app https://github.com/openshift/sti-ruby.git \  
--context-dir=2.0/test/puma-test-app
```

Por fim, é possível criar um aplicativo usando um repositório Git remoto com uma referência específica de ramificação:

```
$ oc new-app https://github.com/openshift/ruby-hello-world.git#beta4
```

Se um fluxo de imagem não for especificado no comando, o `new-app` tenta determinar qual construtor de linguagem usar com base na presença de certos arquivos na raiz do repositório:

Idioma	Arquivos
Ruby	<code>Rakefile Gemfile config.ru</code>
Java EE	<code>pom.xml</code>
Node.js	<code>app.json package.json</code>
PHP	<code>index.php composer.json</code>
Python	<code>requirements.txt config.py</code>
Perl	<code>index.pl cpanfile</code>

Depois que uma linguagem é detectada, o comando `new-app` procura pelas tags de fluxo de imagem que são compatíveis com a linguagem detectada ou um fluxo de imagem que corresponda ao nome da linguagem detectada.

Crie um arquivo de definição de recurso JSON usando o parâmetro `-o json` e o redirecionamento de saída:

```
$ oc -o json new-app php~http://services.lab.example.com/app \
> --name=myapp > s2i.json
```

Esse arquivo de definição JSON cria uma lista de recursos. O primeiro recurso é o fluxo de imagem:

```
...output omitted...
{
    "kind": "ImageStream", ❶
    "apiVersion": "image.openshift.io/v1",
    "metadata": {
        "name": "myapp", ❷
        "creationTimestamp": null
        "labels": {
            "app": "myapp"
        },
        "annotations": {
            "openshift.io/generated-by": "OpenShiftNewApp"
        }
    },
    "spec": {
        "lookupPolicy": {
            "local": false
        }
    },
}
```

```
"status": {  
    "dockerImageRepository": ""  
},  
...output omitted...
```

- ➊ Defina um tipo de recurso de fluxo de imagem.
- ➋ Dê o nome myapp ao fluxo de imagem.

A configuração de compilação (bc) é responsável pela definição dos parâmetros e gatilhos que são executados para transformar o código-fonte em uma imagem executável. O **BuildConfig** (BC) é o segundo recurso e o exemplo a seguir oferece uma visão geral dos parâmetros usados pelo OpenShift para criar uma imagem executável.

```
...output omitted...  
{  
    "kind": "BuildConfig", ①  
    "apiVersion": "build.openshift.io/v1",  
    "metadata": {  
        "name": "myapp", ②  
        "creationTimestamp": null,  
        "labels": {  
            "app": "myapp"  
        },  
        "annotations": {  
            "openshift.io/generated-by": "OpenShiftNewApp"  
        }  
    },  
    "spec": {  
        "triggers": [  
            {  
                "type": "GitHub",  
                "github": {  
                    "secret": "S5_4BZpPabM6KrIuPBvI"  
                }  
            },  
            {  
                "type": "Generic",  
                "generic": {  
                    "secret": "3q8K8JNDoRzhjoz1KgMz"  
                }  
            },  
            {  
                "type": "ConfigChange"  
            },  
            {  
                "type": "ImageChange",  
                "imageChange": {}  
            }  
        ],  
        "source": {  
            "type": "Git",  
            "git": {  
...
```

```
        "uri": "http://services.lab.example.com/app" ③
    },
},
"strategy": {
    "type": "Source", ④
    "sourceStrategy": {
        "from": {
            "kind": "ImageStreamTag",
            "namespace": "openshift",
            "name": "php:7.3" ⑤
        }
    }
},
"output": {
    "to": {
        "kind": "ImageStreamTag",
        "name": "myapp:latest" ⑥
    }
},
"resources": {},
"postCommit": {},
"nodeSelector": null
},
"status": {
    "lastVersion": 0
}
},
...output omitted...
```

- ① Defina um tipo de recurso de `BuildConfig`.
- ② Dê ao `BuildConfig` o nome `myapp`.
- ③ Defina o endereço para o repositório Git do código-fonte.
- ④ Defina a estratégia para usar o S2I.
- ⑤ Defina a imagem do construtor como o fluxo de imagem `php:7.3`.
- ⑥ Dê ao fluxo de imagem de saída o nome `myapp:latest`.

O terceiro recurso é o objeto de implantação que é responsável pela personalização do processo de implantação no OpenShift. Ele pode incluir parâmetros e gatilhos que são necessários para criar novas instâncias de contêiner e são traduzidos em um controlador de replicação do Kubernetes. Alguns dos recursos oferecidos pelo `Deployment` incluem:

- Estratégias personalizadas pelo usuário para fazer a transição entre implantações existentes para novas implantações.
- Ter o máximo de réplicas ativas definidas como desejadas e possíveis.
- O dimensionamento da replicação depende dos tamanhos dos conjuntos de réplicas antigos e novos.

```

...output omitted...
{
    "kind": "Deployment", ①
    "apiVersion": "apps/v1",
    "metadata": {
        "name": "myapp", ②
        "creationTimestamp": null,
        "labels": {
            "app": "myapp",
            "app.kubernetes.io/component": "myapp",
            "app.kubernetes.io/instance": "myapp"
        },
        "annotations": {
            "image.openshift.io/triggers": "[{\"from\": {\"kind\": \"ImageStreamTag\", \"name\": \"myapp:latest\"}, \"fieldPath\": \"spec.template.spec.containers[?(@.name==\\\\\"myapp\\\\\").image\"]\"}], ③
                \"openshift.io/generated-by\": \"OpenShiftNewApp\""
        }
    },
    "spec": {
        "replicas": 1,
        "selector": {
            "matchLabels": {
                "deployment": "myapp"
            }
        },
        "template": {
            "metadata": {
                "creationTimestamp": null,
                "labels": {
                    "deployment": "myapp" ④
                },
                "annotations": {
                    "openshift.io/generated-by": "OpenShiftNewApp"
                }
            },
            "spec": {
                "containers": [
                    {
                        "name": "myapp", ⑤
                        "image": " ", ⑥
                        "ports": [
                            {
                                "containerPort": 8080,
                                "protocol": "TCP"
                            },
                            {
                                "containerPort": 8443,
                                "protocol": "TCP"
                            }
                        ],
                        "resources": {}
                    }
                ]
            }
        }
    }
}

```

```

        ],
    },
    "strategy": {}
},
"status": {}
},
...output omitted...

```

- ➊ Defina um tipo de recurso de Deployment.
- ➋ Dê ao Deployment o nome myapp.
- ➌ Um acionador de alteração de imagem causa a criação de uma nova implantação toda vez que uma nova versão da imagem myapp:latest estiver disponível no repositório.
- ➍ Um acionador de alteração de configuração faz com que uma nova implantação seja criada sempre que o template de controlador de replicação for alterado.
- ➎ Define a imagem do contêiner a ser implantada: myapp:latest.
- ➏ Especifica as portas de contêiner. Um acionador de alteração de configuração faz com que uma nova implantação seja criada sempre que o template de controlador de replicação for alterado.

O último item é o serviço, já abordado em capítulos anteriores:

```

...output omitted...
{
  "kind": "Service",
  "apiVersion": "v1",
  "metadata": {
    "name": "myapp",
    "creationTimestamp": null,
    "labels": {
      "app": "myapp"
      "app.kubernetes.io/component": "myapp",
      "app.kubernetes.io/instance": "myapp"
    },
    "annotations": {
      "openshift.io/generated-by": "OpenShiftNewApp"
    }
  },
  "spec": {
    "ports": [
      {
        "name": "8080-tcp",
        "protocol": "TCP",
        "port": 8080,
        "targetPort": 8080
      },
      {
        "name": "8443-tcp",
        "protocol": "TCP",
        "port": 8443,
        "targetPort": 8443
      }
    ]
  }
}

```

```

        "targetPort": 8443
    }
],
"selector": {
    "deployment": "myapp"
}
},
"status": {
    "loadBalancer": {}
}
}
}

```

**nota**

Por padrão, o comando `oc new-app` não cria uma rota. Você pode criar uma rota depois de criar o aplicativo. Contudo, uma rota é automaticamente criada ao usar o console da web, já que ele usa um template.

Depois de criar um novo aplicativo, o processo de compilação é iniciado. Use o comando `oc get builds` para ver uma lista de compilações de aplicativos:

```
$ oc get builds
NAME          TYPE      FROM      STATUS      STARTED      DURATION
php-helloworld-1  Source   Git@9e17db8  Running  13 seconds ago
```

O OpenShift permite visualizar os logs de compilação. O seguinte comando mostra as últimas linhas do log de compilação:

```
$ oc logs build/myapp-1
```

**Importante**

Se a compilação ainda não estiver com o status `Running`, ou caso o OpenShift ainda não tenha implantado o pod `s2i-build`, o comando acima retorna um erro. Aguarde alguns minutos e tente novamente.

Acione uma nova com build com o comando `oc start-build build_config_name`:

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

```
$ oc start-build myapp
build "myapp-2" started
```

Relação entre criação e implantação

O pod `BuildConfig` é responsável por criar imagens no OpenShift e por enviá-las ao registro interno do contêiner. Qualquer código-fonte ou atualização de conteúdo normalmente requer uma nova compilação para garantir que a imagem seja atualizada.

O pod **Deployment** é responsável por implantar pods no OpenShift. O resultado de uma execução do pod **Deployment** é a criação de pods com as imagens implantadas no registro interno do contêiner. Qualquer pod existente em execução pode ser destruído, dependendo de como o recurso **Deployment** está definido.

Os recursos **BuildConfig** e **Deployment** não interagem diretamente. O recurso **BuildConfig** cria ou atualiza uma imagem de contêiner. O **Deployment** reage a essa nova imagem ou evento de imagem atualizada e cria pods a partir da imagem de contêiner.



Referências

Build do Source-to-Image (S2I)

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/cicd/builds#builds-strategy-s2i-build_understanding-image-builds

Repositório GitHub S2I

<https://github.com/openshift/source-to-image>

► Exercício Guiado

Criação de um aplicativo conteinerizado com Source-to-Image

Neste exercício, você criará um aplicativo a partir do código-fonte e implantará o aplicativo em um cluster do OpenShift.

Resultados

Você deverá ser capaz de:

- Compilar um aplicativo a partir de códigos-fonte usando a interface de linha de comando do OpenShift.
- Verificar a implantação bem-sucedida do aplicativo usando a interface de linha de comando do OpenShift.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Execute o seguinte comando para fazer download dos arquivos de laboratório relevantes e configurar o ambiente:

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

Instruções

- 1. Inspecione o código-fonte PHP do aplicativo de amostra e crie e envie um novo branch chamado `s2i` para uso durante este exercício.
- 1.1. Digite seu clone local do repositório `git D0180-apps` e faça check-out do branch `master` do repositório do curso para garantir que você inicie este exercício em um estado que tenha boas condições:

```
[student@workstation openshift-s2i]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 1.2. Crie um novo branch para salvar as alterações feitas durante este exercício:

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

- 1.3. Revise o código-fonte PHP do aplicativo dentro da pasta `php-helloworld`.

Abra o arquivo `index.php` na pasta `/home/student/D0180-apps/php-helloworld`:

```
<?php  
print "Hello, World! php version is " . PHP_VERSION . "\n";  
?>
```

O aplicativo implementa uma resposta simples que retorna a versão PHP que está sendo executada.

► 2. Prepare o ambiente de laboratório.

- 2.1. Carregue a configuração do ambiente de sala de aula.

Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

```
[student@workstation D0180-apps]$ source /usr/local/etc/ocp4.config
```

- 2.2. Faça login no cluster do OpenShift.

```
[student@workstation D0180-apps]$ oc login -u ${RHT_OCP4_DEV_USER} -p \  
> ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}  
Login successful  
...output omitted...
```

- 2.3. Crie um novo projeto que contenha o nome de usuário do desenvolvedor RHOCP para os recursos que você criar durante este exercício:

```
[student@workstation D0180-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-s2i
```

► 3. Crie um novo aplicativo PHP usando Source-to-Image do diretório `php-helloworld` usando o branch `s2i` que você criou na etapa anterior na sua bifurcação do repositório Git DO180-apps.

- 3.1. Use o comando `oc new-app` para criar um aplicativo PHP.



Importante

O exemplo a seguir usa o sinal de número (#) para selecionar um branch específico do repositório git. Nesse caso, o branch `s2i` criado na etapa anterior.

```
[student@workstation D0180-apps]$ oc new-app php:7.3 --name=php-helloworld \  
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#s2i \  
> --context-dir php-helloworld
```

- 3.2. Aguarde até que a compilação seja concluída e o aplicativo seja implantado. Verifique se o processo de compilação começa com o comando `oc get pods`.

```
[student@workstation openshift-s2i]$ oc get pods
NAME             READY   STATUS    RESTARTS   AGE
php-helloworld-1-build   1/1     Running   0          5s
```

- 3.3. Examine os logs para essa compilação. Use o nome do pod de compilação para essa compilação, `php-helloworld-1-build`.

```
[student@workstation D0180-apps]$ oc logs --all-containers \
> -f php-helloworld-1-build
...output omitted...

Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-...
php@sha256:3206...37b4
Adding transient rw bind mount for /run/secrets/rhsm
STEP 1: FROM image-registry.openshift-image-registry.svc:5000...

...output omitted...

STEP 8: RUN /usr/libexec/s2i/assemble

...output omitted...

Pushing image .../php-helloworld:latest ...
Getting image source signatures
...output omitted...

Writing manifest to image destination
Storing signatures
Successfully pushed .../php-
helloworld@sha256:3f1cdb278548c7f24429e2469c51ae35482d54e2616d596ab6fb59d6b432c454
Push successful
Cloning "https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps" ...
Commit: 9a042f7e3650ef38ad07af83b74f57c7a7d1820c (Added start up script)

...output omitted...
```

O clone do repositório Git é a primeira etapa da compilação. Em seguida, o processo Source-to-Image criou uma nova imagem chamada `${RHT_OCP4_DEV_USER} - s2i/php-helloworld: latest`. A última etapa no processo de compilação é enviar essa imagem ao registro privado do OpenShift.

- 3.4. Analise o Deployment desse aplicativo:

```
[student@workstation D0180-apps]$ oc describe deployment/php-helloworld
Name:           php-helloworld
Namespace:      ${RHT_OCP4_DEV_USER}-s2i
CreationTimestamp: Tue, 30 Mar 2021 12:54:59 -0400
Labels:         app=php-helloworld
                app.kubernetes.io/component=php-helloworld
                app.kubernetes.io/instance=php-helloworld
```

```

Annotations:           deployment.kubernetes.io/revision: 2
                      image.openshift.io/triggers:
                        [{"from":{"kind":"ImageStreamTag","name":"php-
helloworld:latest"},"fieldPath":"spec.template.spec.containers[?(@.name==\"php-
helloworld\")]..."}
                         openshift.io/generated-by: OpenShiftNewApp
Selector:             deployment=php-helloworld
Replicas:              1 desired | 1 updated | 1 total | 1 available | 0
                       unavailable
StrategyType:         RollingUpdate
MinReadySeconds:      0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:               deployment=php-helloworld
  Annotations:          openshift.io/generated-by: OpenShiftNewApp
  Containers:
    php-helloworld:
      Ports:               8080/TCP, 8443/TCP
      Host Ports:          0/TCP, 0/TCP
      Environment:         <none>
      Mounts:              <none>
      Volumes:             <none>
  Conditions:
    Type     Status  Reason
    ----   -----
    Available  True   MinimumReplicasAvailable
    Progressing  True   NewReplicaSetAvailable
OldReplicaSets: <none>
NewReplicaSet:  php-helloworld-6f5d4c47ff (1/1 replicas created)
...output omitted...

```

3.5. Adicione a rota para testar o aplicativo:

```
[student@workstation D0180-apps]$ oc expose service php-helloworld \
> --name ${RHT_OCP4_DEV_USER}-helloworld
route.route.openshift.io/${RHT_OCP4_DEV_USER}-helloworld exposed
```

3.6. Encontre a URL associada à nova rota:

```
[student@workstation D0180-apps]$ oc get route -o jsonpath='{..spec.host}{"\n"}'
${RHT_OCP4_DEV_USER}-helloworld-${RHT_OCP4_DEV_USER}-s2i.
${RHT_OCP4_WILDCARD_DOMAIN}
```

3.7. Teste o aplicativo enviando uma solicitação GET HTTP à URL que você obteve na etapa anterior:

```
[student@workstation D0180-apps]$ curl -s \
> ${RHT_OCP4_DEV_USER}-helloworld-${RHT_OCP4_DEV_USER}-s2i.\
> ${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.29
```

- 4. Explore a inicialização de compilações de aplicativos alterando o aplicativo em seu repositório Git e executando os comandos apropriados para iniciar uma nova compilação Source-to-Image.

4.1. Digite o diretório do código-fonte.

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

4.2. Edite o arquivo `index.php` conforme mostrado abaixo:

```
<?php  
print "Hello, World! php version is " . PHP_VERSION . "\n";  
print "A change is a coming!\n";  
?>
```

Salve o arquivo.

4.3. Confirme as alterações e envie o código de volta ao repositório Git remoto:

```
[student@workstation php-helloworld]$ git add .  
[student@workstation php-helloworld]$ git commit -m 'Changed index page contents.'  
[s2i b1324aa] changed index page contents  
 1 file changed, 1 insertion(+)  
[student@workstation php-helloworld]$ git push origin s2i  
...output omitted...  
Counting objects: 7, done.  
Delta compression using up to 2 threads.  
Compressing objects: 100% (3/3), done.  
Writing objects: 100% (4/4), 417 bytes | 0 bytes/s, done.  
Total 4 (delta 1), reused 0 (delta 0)  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps  
 f7cd896..b1324aa s2i -> s2i
```

4.4. Inicie um novo processo de compilação Source-to-Image e aguarde até que ele seja compilado e implantado:

```
[student@workstation php-helloworld]$ oc start-build php-helloworld  
build.build.openshift.io/php-helloworld-2 started  
[student@workstation php-helloworld]$ oc get pods  
NAME READY STATUS RESTARTS AGE  
php-helloworld-1-build 0/1 Completed 0 5m7s  
php-helloworld-2-build 0/1 Completed 0 43s  
...output omitted...  
[student@workstation php-helloworld]$ oc logs php-helloworld-2-build -f  
...output omitted...  
  
Successfully pushed .../php-helloworld:latest@sha256:74e757a4c0edaeda497dab7...  
Push successful
```

**nota**

Os logs podem levar alguns segundos para estarem disponíveis depois que a compilação for iniciada. Se o comando anterior falhar, aguarde um pouco e tente novamente.

- 4.5. Depois que a segunda construção for concluída, use o comando `oc get pods` para verificar se a nova versão do aplicativo está sendo executada.

```
[student@workstation php-helloworld]$ oc get pods
NAME           READY   STATUS    RESTARTS   AGE
php-helloworld-1-build  0/1     Completed  0          11m
php-helloworld-1-deploy 0/1     Completed  0          10m
php-helloworld-2-build  0/1     Completed  0          45s
php-helloworld-2-deploy 0/1     Completed  0          16s
php-helloworld-2-wq9wz  1/1     Running   0          13s
```

- 4.6. Teste se o aplicativo transmite o novo conteúdo:

```
[student@workstation php-helloworld]$ curl -s \
> ${RHT_OCP4_DEV_USER}-helloworld-${RHT_OCP4_DEV_USER}-s2i.\
> ${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.29
A change is a coming!
```

Encerramento

Na workstation, execute o script `lab openshift-s2i finish` para concluir esse laboratório.

```
[student@workstation php-helloworld]$ lab openshift-s2i finish
```

Isso conclui o exercício orientado.

Criação de aplicativos com o console da web do OpenShift

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Criar um aplicativo com o console da web do OpenShift.
- Gerenciar e monitorar o ciclo de compilação de um aplicativo.
- Examinar os recursos de um aplicativo.

Acesso ao console da web do OpenShift.

O console da web do OpenShift permite aos usuários executar muitas das mesmas tarefas que podem executar como cliente da linha de comando do OpenShift. Você pode criar projetos, adicionar aplicativos a projetos, visualizar recursos de aplicativos e manipular configurações de aplicativos conforme necessário. O console da web do OpenShift é executado como um ou mais pods, cada pod rodando em um plano de controle.

Embora o cliente de linha de comando seja mais versátil do que o console web, a representação visual dos conceitos no OpenShift pode ser um aprendizado útil.

O console da web é executado em um navegador da web.

A URL padrão está no formato `https://console-openshift-console.{wildcard DNS domain for the RHOC cluster}`. Por padrão, o OpenShift gera um certificado autoassinado para o console da web. Você deve confiar nesse certificado para obter acesso.

O console da web usa uma API REST para se comunicar com o cluster do OpenShift. Por padrão, o ponto de extremidade da API REST é acessado com um nome DNS diferente e um certificado autoassinado. Você também deve confiar nesse certificado para o ponto de extremidade da API REST.

Depois que você tiver confiado nos dois certificados do OpenShift, o console exige autenticação para continuar.

Gerenciamento de projetos

Após o login bem-sucedido, a página Home > Projects exibe uma lista de projetos que você pode acessar. Nesta página você pode criar, editar ou excluir um projeto.

Se você tiver permissão para exibir métricas de clusters, será redirecionado para a página Home > Overview. Essa página mostra informações gerais e métricas sobre o cluster. O item de menu Overview é oculto de usuários sem autoridade para exibir métricas de cluster.

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

Figura 6.9: Página inicial do console da web do OpenShift

O ícone de reticências no final de cada linha fornece um menu com ações do projeto. Selecione a entrada apropriada para editar ou excluir o projeto.

Se você clicar em um link de projeto nesta visualização, será redirecionado para a página **Project Status**, que mostra todos os aplicativos criados dentro desse espaço do projeto.

Navegação no console da web

Há um menu de navegação do lado esquerdo do console da web. O menu inclui duas perspectivas: **Administrator** e **Developer**. Cada item no menu se expande para fornecer acesso a um conjunto de funções de gerenciamento relacionadas. Quando a perspectiva do **Administrator** estiver selecionada, esses itens serão os seguintes:

Página inicial

O menu inicial permite que os usuários acessem rapidamente os projetos e os recursos. Nesse menu, você pode navegar e gerenciar projetos, pesquisar ou explorar recursos de cluster e inspecionar eventos de cluster.

Operadores

O OperatorHub é um catálogo que permite a você descobrir, pesquisar e instalar operadores no cluster. Depois de instalar um operador, você pode usar a opção **Installed Operators** para gerenciar o operador ou pesquisar outros operadores instalados.



nota

As versões mais recentes do Kubernetes implementam muitos controladores como Operadores. Operadores são componentes plug-in do Kubernetes que podem reagir a eventos de cluster e controlar o estado dos recursos. Os operadores e o CoreOS Operator Framework estão fora do escopo deste documento.

Cargas de trabalho

Essas opções permitem o gerenciamento de vários tipos de recursos do Kubernetes e OpenShift, como pods e implantações. Outras opções avançadas de implantação que são acessíveis a partir desse menu, como mapas de configuração, segredos e tarefas agendadas, estão além do escopo do curso.

Rede

Esse menu contém opções para gerenciar recursos do OpenShift que afetam o acesso a aplicativos, como serviços e rotas, para um projeto. Outras opções para configurar uma entrada ou política de rede do OpenShift estão disponíveis, mas esses tópicos estão fora do escopo deste curso.

Armazenamento

Este menu contém opções para configurar o armazenamento persistente para aplicativos de projeto. Em particular, volumes persistentes e solicitações de volumes persistentes para um projeto são gerenciados no menu **Storage**.

Builds

A opção **Build Configs** exibe uma lista de configurações de compilação do projeto. Clique em um link de configuração de compilação nesta visualização para acessar uma página de visão geral da configuração de compilação especificada. Nessa página, você pode visualizar e editar a configuração de compilação do aplicativo.

A opção **Builds** fornece uma lista de processos de compilação recentes para imagens de contêiner de aplicativo no projeto. Clique no link para uma compilação específica para acessar os logs de compilação para esse processo de compilação específico.

A opção **Image Streams** fornece uma lista de fluxos de imagem definidos no projeto. Clique em uma entrada de fluxo de imagem nesta lista para acessar uma página de visão geral a fim de visualizar e gerenciar esse fluxo de imagem.

Computação

Fornece opções de acesso e gerenciamento de nós de computação do cluster do OpenShift. Nessa página, você também pode configurar verificações de integridade de nó e dimensionamento automático.

Gerenciamento do usuário

Nessa opção, você pode configurar a autenticação e a autorização usando usuários, grupos, funções, vinculações de funções e contas de serviço.

Administration

Fornece opções para gerenciar configurações de cluster e projeto, como cotas de recursos e controles de acesso baseados em função. As funções da seção **Administration** estão fora do escopo deste curso.

Criação de novos aplicativos

Na perspectiva do **Developer**, use **+Add** para selecionar uma forma de criar um novo aplicativo em um projeto do OpenShift. Você pode adicionar um aplicativo do **Developer Catalog**, usando a opção **All services**, o que oferece uma seleção de templates Source-to-Image (S2I), imagens do construtor e gráficos Helm para criar aplicativos específicos da tecnologia. Selecione um modelo desejado e forneça as informações necessárias para implantar o novo aplicativo.

Você não está limitado a implantação de um aplicativo do catálogo. Você também pode implantar um aplicativo usando:

- Uma imagem de contêiner hospedada em um registro de contêiner remoto.

- Um arquivo YAML que especifica os recursos do Kubernetes e do OpenShift a serem criados.
- Uma imagem de construtor usando o código-fonte ou Containerfile de seu próprio repositório git.

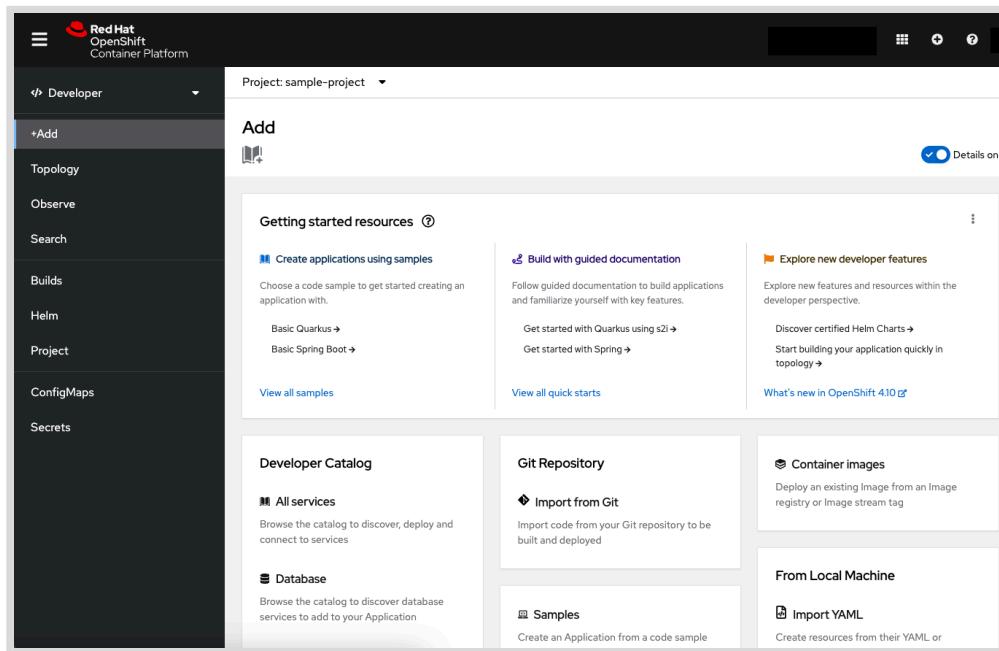


Figura 6.10: Página Developer Catalog do OpenShift

Para criar um aplicativo com um desses métodos, selecione a opção apropriada na página **Add**. Use a opção **Container images** para implantar uma imagem de contêiner existente. Use a opção **Import YAML** para criar os recursos especificados em um arquivo YAML. Use a opção **Import from Git** para implantar seu código-fonte usando uma imagem de construtor ou seu Containerfile. As opções **Database**, **Operator Backed** e **Helm Chart** são atalhos do catálogo.

Gerenciamento de builds de aplicativos

Na perspectiva **Administrator**, clique na opção **Build Configs** do menu **Builds** depois de adicionar um aplicativo Source-to-Image a um projeto. A nova configuração de build está acessível nesta exibição:

Figura 6.11: Página de configurações de compilação do OpenShift

Clique em uma configuração de compilação na lista para visualizar uma página de visão geral da configuração de compilação selecionada. Na página de visão geral, você pode:

- Visualize os parâmetros de configuração da compilação, como o URL do repositório Git do código-fonte.
- Visualize e edite as variáveis de ambiente que estão configuradas no contêiner do construtor, durante um processo de construção do aplicativo.
- Visualize uma lista de compilações de aplicativos recentes e clique em uma compilação selecionada para acessar os logs do processo de criação.

Gerenciamento de implantações de aplicativos

O menu Workloads oferece acesso às implantações no projeto.

Figura 6.12: Menu Workloads do OpenShift

Em cargas de trabalho, você encontra muitos dos constructos discutidos no curso, incluindo pods, implantações e mapas de configuração.

Clique em uma entrada de implantação na lista para visualizar uma página de visão geral da seleção. Na página de visão geral, você pode:

- Visualizar os parâmetros de implantação, como as especificações de uma imagem de contêiner de aplicativo.
- Alterar o número desejado de pods de aplicativo para dimensionar manualmente o aplicativo.
- Visualizar e editar as variáveis de ambiente que estão configuradas no contêiner do aplicativo implantado.
- Visualizar uma lista de pods de aplicativos e clicar em um pod selecionado para acessar os logs desse pod.

Outros recursos do console da web

O console da web permite:

- Gerenciar recursos como cotas de projeto, associação de usuário, segredos e outros recursos avançados.
- Criar solicitações de volume persistente.
- Monitorar compilações, implantações, pods e eventos do sistema.
- Criar pipelines de integração e implantação contínuas.

O uso detalhado dos recursos acima está fora do escopo deste curso.

► Exercício Guiado

Criação de um aplicativo com o console da web

Neste exercício, você criará, compilará e implantará um aplicativo em um cluster do OpenShift usando o console da web do OpenShift.

Resultados

Você deverá ser capaz de criar, compilar e implantar um aplicativo em um cluster do OpenShift usando o console da web.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Obtenha os arquivos do laboratório executando o script do laboratório:

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

O script do laboratório verifica se o cluster do OpenShift está em execução.

Instruções

- 1. Inspecione o código-fonte PHP do aplicativo de amostra e crie e envie um novo branch chamado `console` para uso durante este exercício.
 - 1.1. Digite seu clone local do repositório `git D0180-apps` e faça check-out do branch `master` do repositório do curso para garantir que você inicie este exercício em um estado que tenha boas condições:

```
[student@workstation ~]$ cd ~/D0180-apps  
[student@workstation D0180-apps]$ git checkout master  
...output omitted...
```

- 1.2. Crie um novo branch para salvar as alterações feitas durante este exercício:

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

- 1.3. Revise o código-fonte PHP do aplicativo dentro da pasta `php-helloworld`.
Abra o arquivo `index.php` na pasta `/home/student/D0180-apps/php-helloworld`:

```
<?php  
print "Hello, World! php version is " . PHP_VERSION . "\n";  
?>
```

O aplicativo implementa uma resposta simples que retorna a versão PHP que está sendo executada.

- 2. Abra um navegador da web e navegue até [https://console-openshift-console.\\${RHT_OCP4_WILDCARD_DOMAIN}](https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}) para acessar o console da web do OpenShift. Faça login e crie um novo projeto chamado *youruser-console*.

- 2.1. Carregue a configuração do ambiente de sala de aula.

Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- 2.2. Recupere o valor do domínio curinga específico para o cluster, usando o `$RHT_OCP4_WILDCARD_DOMAIN`

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

- 2.3. Abra o navegador Firefox e navegue até [https://console-openshift-console.\\${RHT_OCP4_WILDCARD_DOMAIN}](https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}) para acessar o console da web do OpenShift. Faça login no console do OpenShift usando suas credenciais.

- 2.4. Crie um novo projeto chamado *youruser-console*. Você pode digitar o valor que preferir nos outros campos.

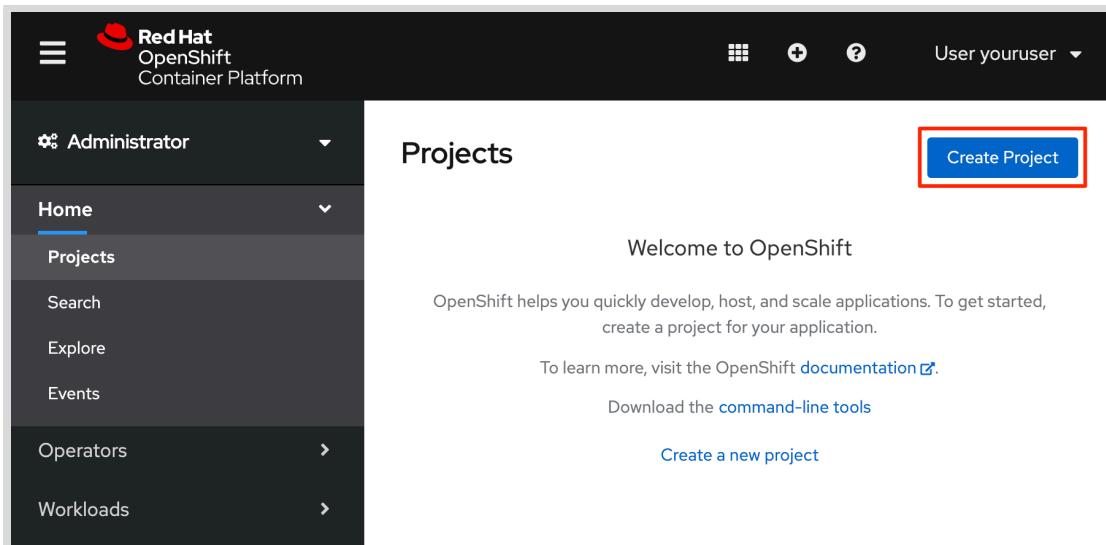


Figura 6.13: Crie um novo projeto

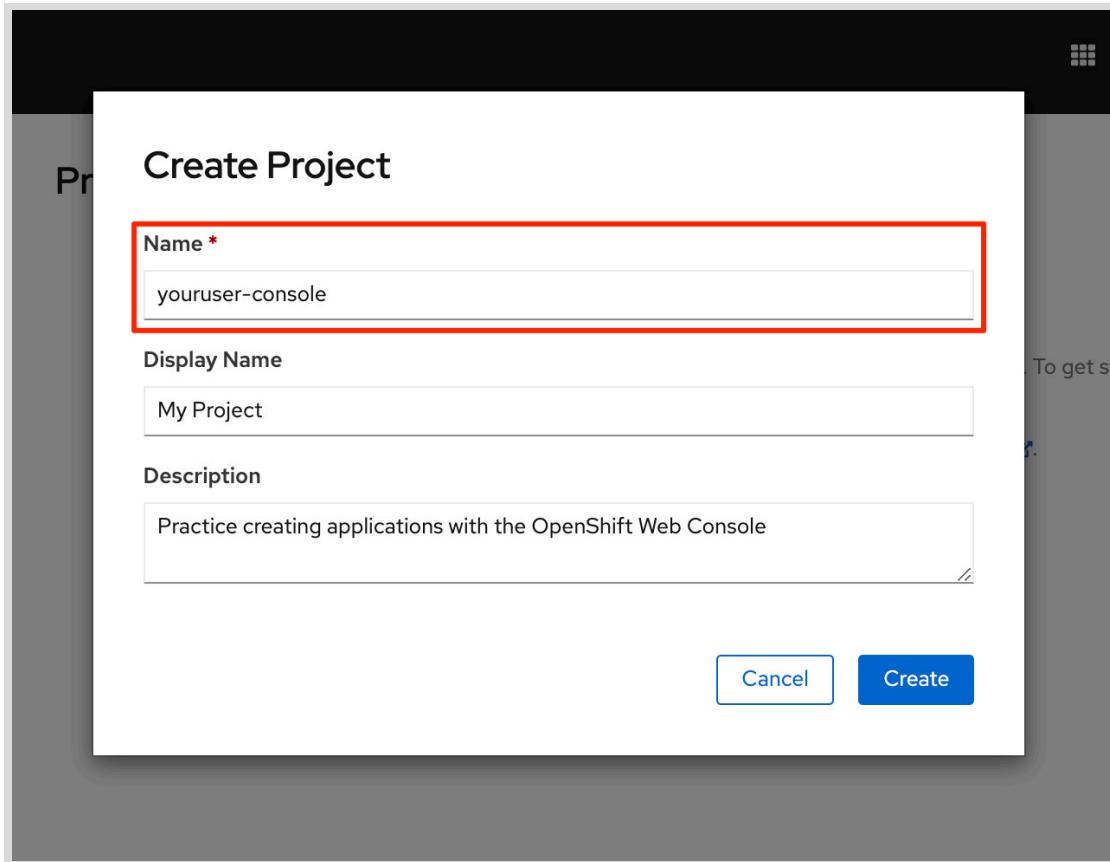


Figura 6.14: Crie um novo projeto

- 2.5. Depois de preencher os campos obrigatórios, clique em **Create** na caixa de diálogo **Create Project** para acessar a página **Project Status** do projeto **youruser-console**:

Details		Status		Activity	
Name	youruser-console	Requester	youruser	Activity	View events
Labels	No labels	Utilization		Ongoing	
		Resource	Usage	There are no ongoing activities.	
		CPU	Not available	Recent Events	
		Memory	Not available	Pause	
		Filesystem	Not available	There are no recent events.	
Inventory					
0 Deployments					
0 Deployment Configs					
0 Stateful Sets					

Figura 6.15: Página Project Status

► 3. Crie o novo aplicativo php-helloworld com um template PHP.

- 3.1. Na lista na parte superior do menu à esquerda, selecione a perspectiva **Developer**.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The top navigation bar displays the Red Hat logo and 'OpenShift Container Platform'. On the left, a sidebar menu is open under the 'Administrator' section, with the 'Developer' option highlighted by a red box. The main content area shows a project named 'PR youruser-console' with the status 'Active'. Below the project name, there are tabs for 'Overview', 'Details', 'YAML', 'Workloads', and 'Role Bindings'. The 'Overview' tab is selected. A table provides details about the project, including 'Name: youruser-console' and 'Status: Active'.

Figura 6.16: Menu suspenso de perspectiva do desenvolvedor

- 3.2. Clique em **+Add** no menu à esquerda. Na página aberta, clique em **All services**, que está na seção **Developer Catalog**, para exibir uma lista de templates de tecnologia.

The screenshot shows the 'Developer Catalog' page within the Red Hat OpenShift interface. The left sidebar is still in 'Developer' mode. The main area is titled 'Developer Catalog' and contains a sub-header: 'Add shared applications, services, event sources, or source-to-image builders to your Project from the developer catalog. Cluster administrators can customize the content made available in the catalog.' Below this, there are two sections: 'All items' and 'All items'. Under 'All items', there are categories: CI/CD, Databases, Languages, Middleware, Other, Type (Builder Images (11), Devfiles (4), Helm Charts (22), Operator Backed (21), Templates (79)). To the right, four specific service templates are listed in a grid:

- .NET** (Builder Images): Build and run .NET 6 applications on UBI 8. More info.
- .NET** (Helm Charts): A Helm chart to build and deploy .NET applications. Provided by Red Hat.
- 3scale APICast API Gateway** (Templates): 3scale's APICast is an NGINX based API gateway used to integrate your internal and...
- Apache HTTP Server** (Templates): An example Apache HTTP Server (httpd) application that serves static content. For more...

Figura 6.17: Página Developer Catalog

- 3.3. Digite **php** no campo **Filter by keyword**.

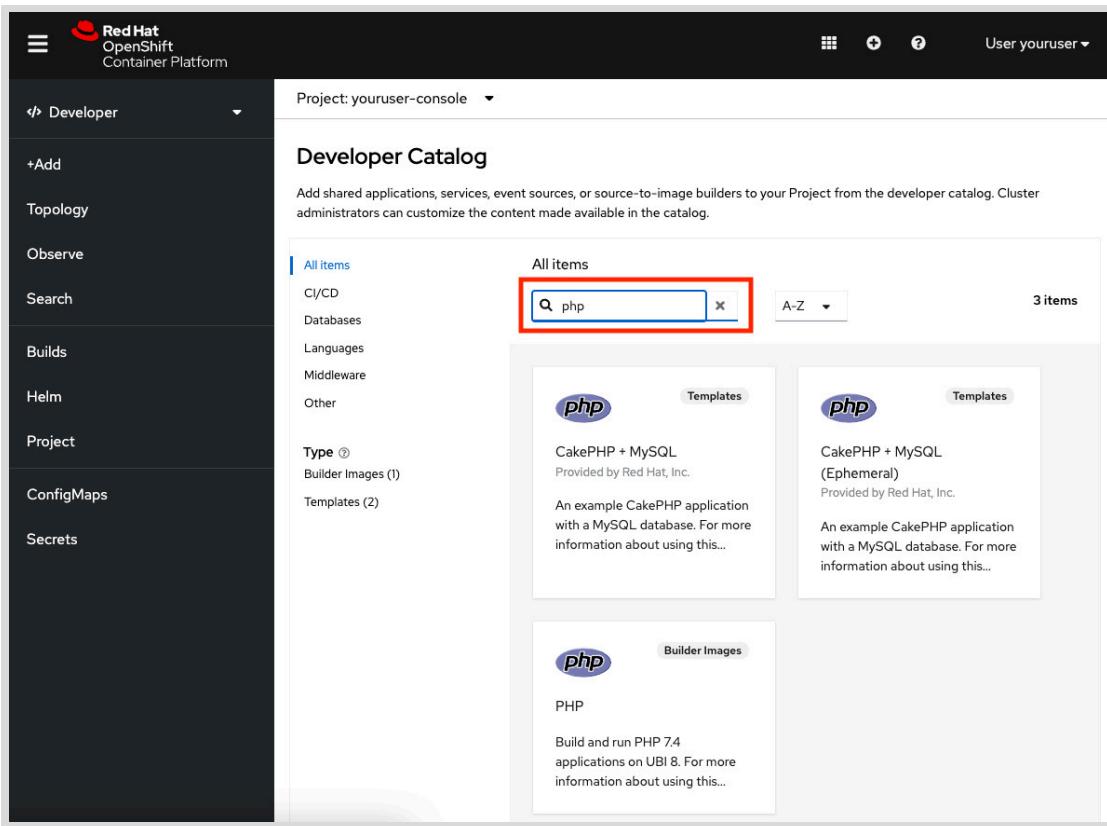


Figura 6.18: Encontrar templates relacionados a PHP

- 3.4. Após a filtragem, clique na imagem de construtor de PHP para exibir a caixa de diálogo PHP. Clique em **Create Application** para exibir a página **Create Source-to-Image Application**.

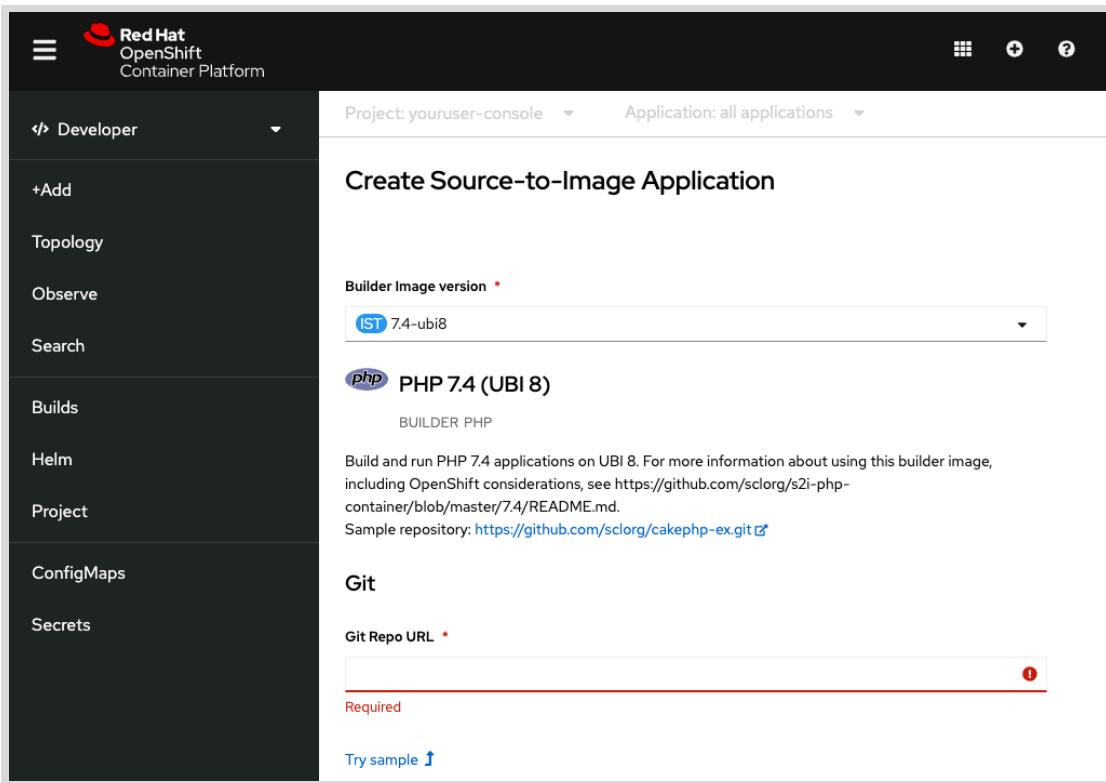


Figura 6.19: Configuração de Source-to-Image para um aplicativo PHP

- 3.5. Altere a **Builder Image Version** para PHP versão 7.4 (UBI 8).

Especifique o local repositório git do código-fonte: <https://github.com/yourgituser/D0180-apps>

Use **Advanced Git Options** para definir o diretório do contexto como `php-helloworld` e o branch `console` para este exercício

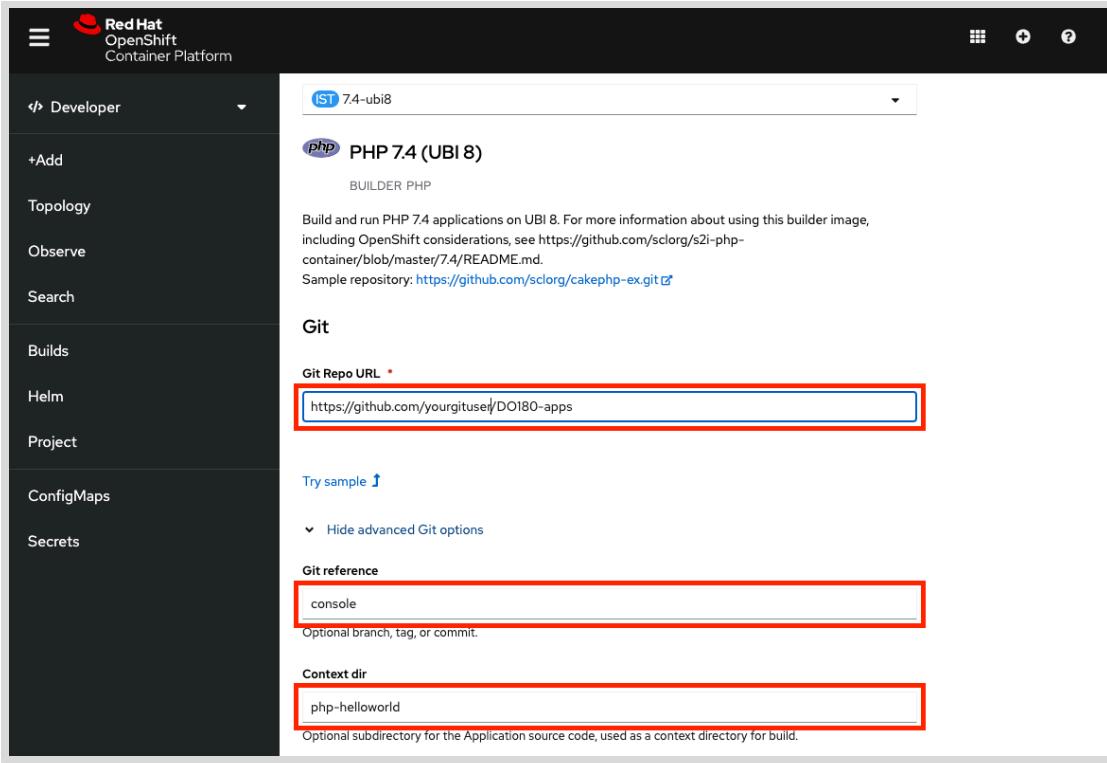


Figura 6.20: Definição de opções Git avançadas para o aplicativo

Digite php-helloworld para o nome do aplicativo e o nome usado para os recursos associados. Selecione **Deployment** como tipo de recurso.

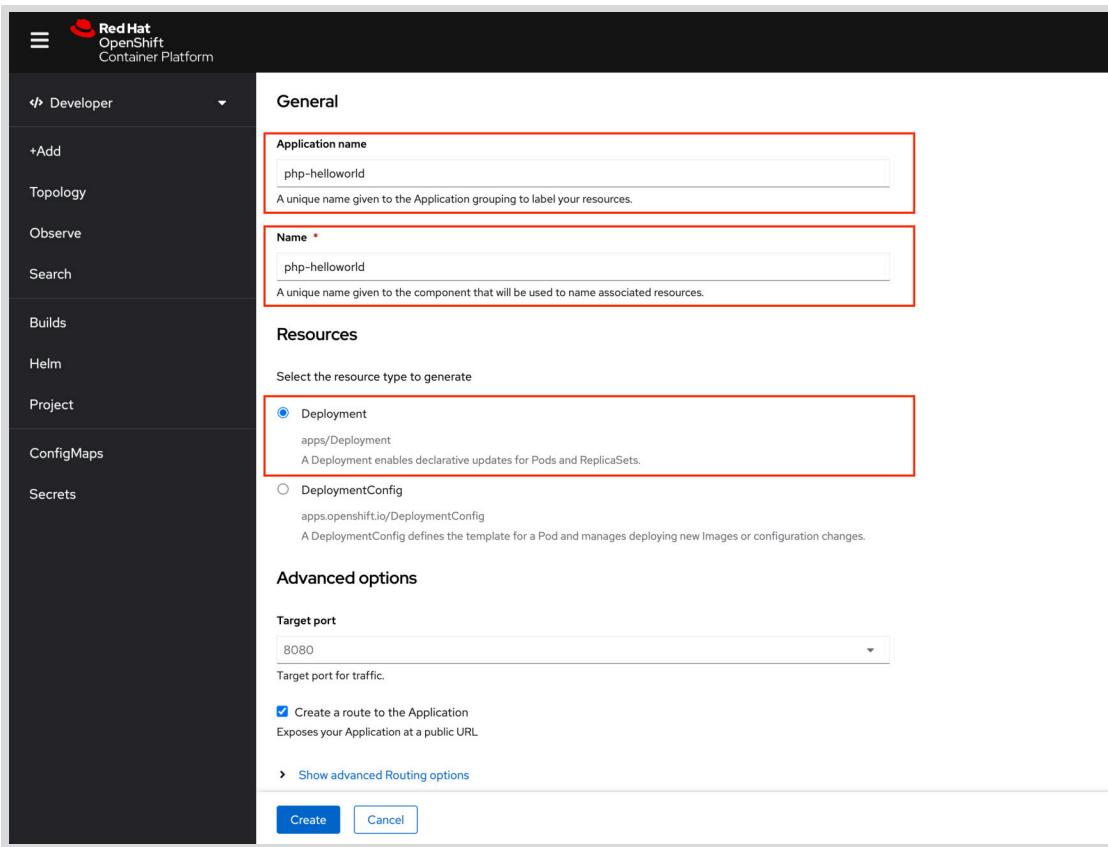


Figura 6.21: Configuração de opções de aplicativo

Role até o final da página e selecione **Create a route to the application**. Clique em **Create** para criar os recursos necessários do OpenShift e do Kubernetes para o aplicativo.

Você é redirecionado à página **Topology**:

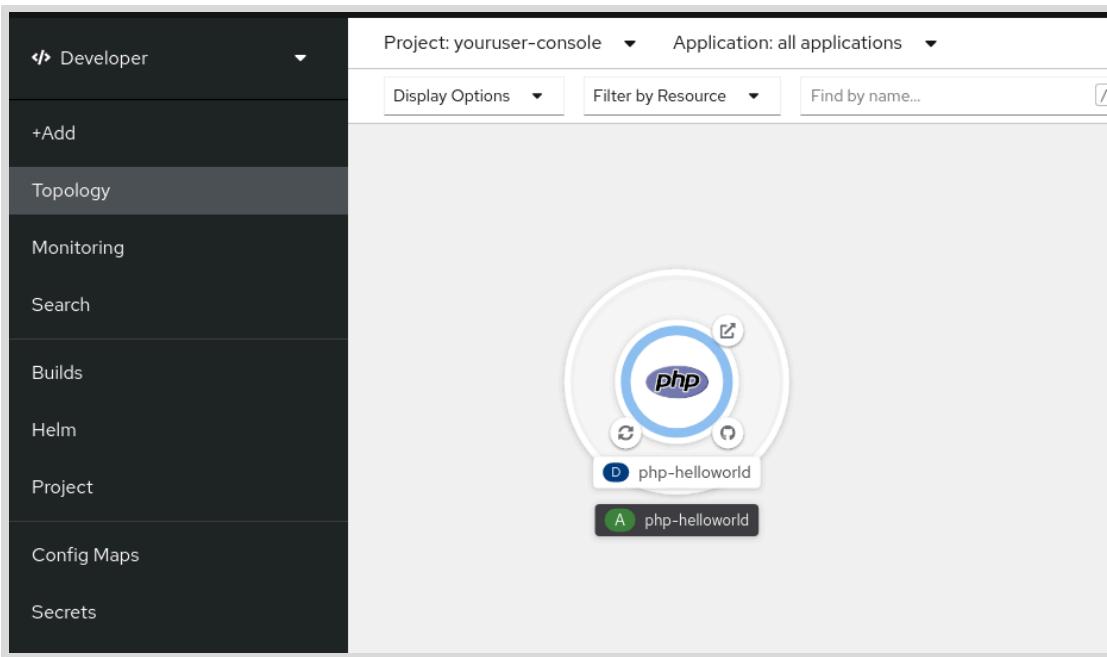


Figura 6.22: Página Topologia

Esta página indica que o aplicativo `php-helloworld` é criado. A anotação D à esquerda do link `php-helloworld` é um acrônimo para Deployment. Esse link redireciona para uma página que contém informações sobre implantação do aplicativo.

3.6. Volte para a perspectiva **Administrator** para o restante do exercício:

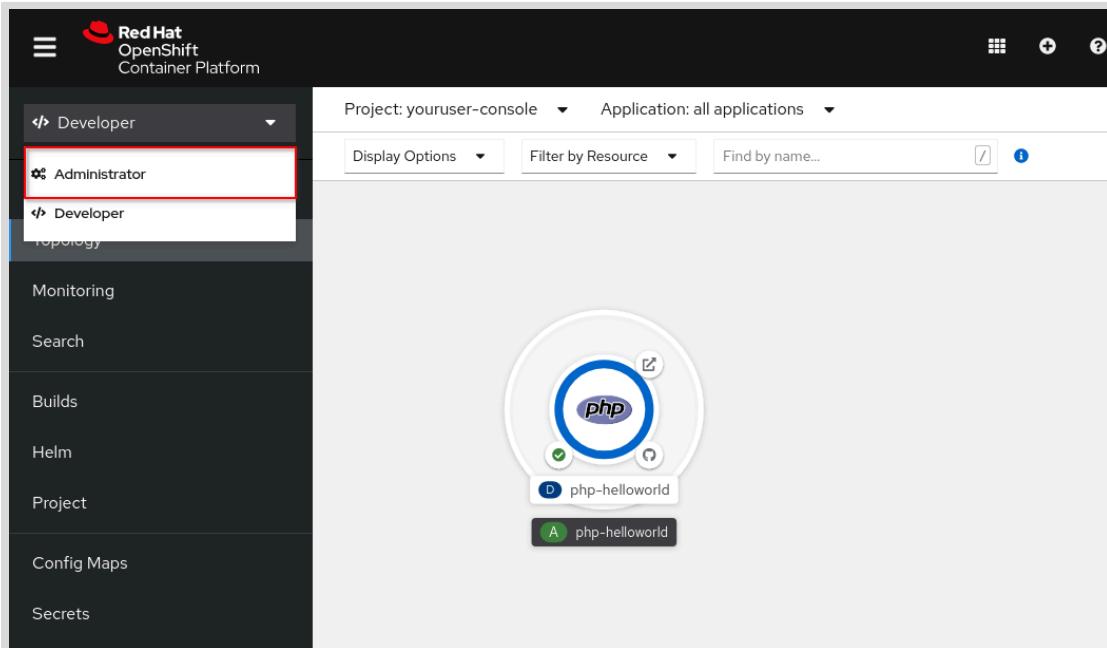


Figura 6.23: Menu suspenso de perspectiva do administrador

- 4. Use a barra de navegação no lado esquerdo do console da web do OpenShift para localizar informações sobre os recursos do OpenShift e Kubernetes do aplicativo:

- Deployments
 - BuildConfig
 - Build Logs
 - Serviço
 - Route
- 4.1. Examine a implantação. Na barra de navegação, clique em **Workloads > Deployments** para exibir uma lista de implementações para o projeto `youruser-console`. Clique no link `php-helloworld` para exibir detalhes de implantação.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar has a dark theme with categories like Events, Operators, Workloads (Pods, Deployments, DeploymentConfigs, StatefulSets, Secrets, ConfigMaps), CronJobs, Jobs, DaemonSets, ReplicaSets, ReplicationControllers, and HorizontalPodAutoscalers. The Networking, Storage, and Builds sections are also listed. The main content area is titled 'Deployment details' for 'php-helloworld'. It shows a circular icon with a blue border and a white center containing the number '1'. Below it, the deployment is named 'php-helloworld' in the 'Namespace' section, which is part of the 'imzziv-console' project. The 'Labels' section lists several key-value pairs: app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php-helloworld, app.kubernetes.io/part-of=php-helloworld, app.openshift.io/runtime=php, and app.openshift.io/runtime-version=7.4-ubi8. To the right of these labels are configuration parameters: 'Update strategy' set to RollingUpdate, 'Max unavailable' set to 25% of 1 pod, 'Max surge' set to 25% greater than 1 pod, 'Progress deadline seconds' set to 600 seconds, and 'Min ready seconds' set to Not configured. At the bottom of the main content area, there's a 'Pod selector' section with a search bar containing 'app=php-helloworld'.

Figura 6.24: Página de detalhes de implantação de aplicativos

Explore as informações disponíveis na aba **Details**. A compilação ainda pode estar em execução quando você acessar essa página, por isso, a implantação pode ainda não ter um valor de 1 pod.

Se você clicar nos ícones de seta para cima e para baixo ao lado do gráfico de rosca que indica o número de pods, poderá dimensionar o aplicativo para cima e para baixo horizontalmente.

- 4.2. Examine a configuração da compilação. Na barra de navegação, clique em **Builds > Build Configs** para exibir uma lista de configurações de build para o projeto `youruser-console`. Clique no link `php-helloworld` para exibir a configuração de build para o aplicativo.

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar has sections for Jobs, DaemonSets, ReplicaSets, ReplicationControllers, and HorizontalPodAutoscalers. Under Networking, Services, Routes, Ingresses, and NetworkPolicies are listed. Storage, Builds, Compute, and User Management sections are also present. The main content area shows a 'BuildConfig details' page for 'BC php-helloworld'. The 'Details' tab is selected. Key information includes:

- Name:** php-helloworld
- Type:** Source
- Namespace:** youruser-console
- Labels:** app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php-helloworld, app.kubernetes.io/part-of=php-helloworld, app.openshift.io/runtime=php, app.openshift.io/runtime-version=7.4-ubi8
- Annotations:** 3 annotations
- Created at:** Feb 25, 2022, 1:14 PM
- Owner:** No owner
- Git repository:** https://github.com/yourgithubuser/DO180-apps
- Git ref:** console
- Context dir:** php-helloworld
- Build from:** IS php:7.4-ubi8
- Output to:** IS php-helloworld:latest
- Run policy:** Serial
- Triggers:** Generic, GitHub, ImageChange, ConfigChange

Figura 6.25: Página de detalhes da configuração de build de aplicativos

Explore as informações disponíveis na aba **Details**. A guia **YAML** permite que você visualize e edite a configuração de build como um arquivo YAML. A guia **Builds** fornece uma lista histórica de compilações junto com um link para mais informações sobre cada compilação. A guia **Environment** permite visualizar e editar variáveis de ambiente para o ambiente de compilação do aplicativo. A guia **Events** exibe uma lista de eventos e metadados relacionados à compilação.

- 4.3. Examine os logs da build Source-to-Image do aplicativo. No menu **Builds**, clique em **Builds** para exibir uma lista de builds recentes do projeto **youruser-console**.

Clique no link **php-helloworld-1** para acessar informações sobre a primeira build do aplicativo **php-helloworld**:

capítulo 6 | Implantação de aplicativos conteinerizados no OpenShift

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, there's a sidebar with various project navigation options like Deployments, DeploymentConfigs, StatefulSets, Secrets, ConfigMaps, CronJobs, Jobs, DaemonSets, ReplicaSets, ReplicationControllers, and HorizontalPodAutoscalers. Under the 'Builds' section, 'Builds' is selected. The main content area is titled 'Build details' for 'php-helloworld-1'. It shows the build status as 'Complete', the namespace as 'youruser-console', and the type as 'Source'. It also displays the Git repository URL and the build configuration details. The 'Logs' tab is visible at the bottom of the main content area.

Figura 6.26: Página de detalhes de uma build de aplicativos

Explore as informações disponíveis na aba Details. Em seguida, clique na guia Logs. Uma caixa de texto rolável contém a saída do processo de compilação:

The screenshot shows the same Red Hat OpenShift Container Platform interface as Figure 6.26, but with the 'Logs' tab selected in the 'Build details' section. The terminal window displays the build log output, which includes messages about copying blobs, writing manifests, and pushing the image to the registry. The log ends with a successful push message.

```

63 lines
43 Copying blob sha256:7f1fd656af6e7b6afb2b3523a01c717b13d94bd9dede6c1f0061ed9e70b22d65
44 Copying config sha256:98a358743a0541f78e13a5a14a9aad4fd4acd3c505ab5d057fc274581887212
45 Writing manifest to image destination
46 Storing signatures
47 --> 98a358743a0
48 Successfully tagged temp.builder.openshift.io/jmzzjv-console/php-helloworld-1:3454a6ca
49 98a358743a0541f78e13a5a14a9aad4fd4acd3c505ab5d057fc274581887212
50
51 Pushing image image-registry.openshift-image-registry.svc:5000/jmzzjv-console/php-helloworld:latest ...
52 Getting image source signatures
53 Copying blob sha256:7f1fd656af6e7b6afb2b3523a01c717b13d94bd9dede6c1f0061ed9e70b22d65
54 Copying blob sha256:5dcbd60ea6b60326f98e2b49d6ebcb771d4fb70c6297ddf2d7dede6692df6e
55 Copying blob sha256:79a56ba04a301eb949644bc29f18b1879bf385091ef1eb8068a0f5828db863
56 Copying blob sha256:867113e1c57d3106acae2383f9bbfe1c45a26eacb03ec82786a494e15956c3
57 Copying blob sha256:12b8b1afe30f6fe8a264840ad3f250b82b335b9ebdc922db57b48836cee08c8
58 Copying blob sha256:aad543859364662ddb264ad5752fd9449d47410b9ef0278463c0a9c578b79c6
59 Copying config sha256:98a358743a0541f78e13a5a14a9aad4fd4acd3c505ab5d057fc274581887212
60 Writing manifest to image destination
61 Storing signatures
62 Successfully pushed image image-registry.openshift-image-registry.svc:5000/jmzzjv-console/php-helloworld@sha256:62
63 Push successful

```

Figura 6.27: Logs para uma compilação de aplicativo

Quando o Podman cria uma imagem de contêiner, uma saída semelhante é observada em comparação à saída mostrada no navegador.

- 4.4. Localize informações para o serviço `php-helloworld` do aplicativo. Na barra de navegação, clique em **Networking > Services** para exibir uma lista de serviços para o projeto `youruser-console`. Clique no link `php-helloworld` para exibir as informações associadas ao serviço do aplicativo:

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar has a dark theme with white text. The 'Networking' section is expanded, and 'Services' is selected. The main content area shows a service named 'php-helloworld' under the project 'youruser-console'. The 'Details' tab is active. The 'Service details' section includes fields for 'Name' (php-helloworld), 'Namespace' (NS youruser-console), and 'Labels' (app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php-helloworld, app.kubernetes.io/part-of=php-helloworld, app.openshift.io/runtime=php, app.openshift.io/runtime-version=7.4-ubi8). The 'Service routing' section shows the 'Hostname' as 'youruser-console' and 'Accessible within the cluster only'. The 'Service address' section shows a 'Cluster IP' of '172.30.215.25' and 'Accessible within the cluster only'. The 'Service port mapping' section lists two entries: '8080-tcp' (Service Port 8080, Pod Port 8080, Protocol TCP) and '8443-tcp' (Service Port 8443, Pod Port 8443, Protocol TCP).

Figura 6.28: Página de detalhes de serviço

Explore as informações disponíveis na aba **Details**. A guia **YAML** permite que você visualize e edite a configuração do serviço como um arquivo YAML. A guia **Pods** exibe a lista atual de pods que fornecem o serviço de aplicativo.

- 4.5. Localize informações de rota externa para o aplicativo. Na barra de navegação, clique em **Networking > Routes** para exibir uma lista de rotas configuradas para o projeto `youruser-console`. Clique no link `php-helloworld` para exibir informações associadas à rota do aplicativo:

The screenshot shows the Red Hat OpenShift Container Platform web interface. On the left, there is a navigation sidebar with sections like Home, Operators, Workloads, Networking, Storage, Builds, Compute, User Management, and Administration. Under Networking, the 'Routes' option is selected. The main content area is titled 'Route details' for a route named 'php-helloworld'. It shows the following details:

- Name:** php-helloworld
- Namespace:** youruser-console
- Labels:** app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php-helloworld, app.kubernetes.io/part-of=php-helloworld, app.openshift.io/runtime=php, app.openshift.io/runtime-version=7.4-ubi8
- Annotations:** 1 annotation
- Service:** php-helloworld
- Location:** https://php-helloworld-youruser-console.apps.cluster.lab.example.com
- Status:** Accepted
- Host:** php-helloworld-youruser-console.apps.cluster.lab.example.com
- Path:** -
- Router canonical hostname:** router-default.apps.cluster.lab.example.com

Figura 6.29: Página de detalhes da rota

Explore as informações disponíveis na aba Details. O campo Location fornece um link para a rota externa do aplicativo, `https://php-helloworld-$\{RHT_OCP4_DEV_USER\}-console.\$\{RHT_OCP4_WILDCARD_DOMAIN\}`. Clique no link para acessar o aplicativo em uma nova guia:

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

Figura 6.30: Resultados iniciais do aplicativo PHP

- 5. Modifique o código do aplicativo, faça commit da alteração, envie o código ao repositório Git remoto e acione uma nova compilação do aplicativo.

- 5.1. Digite o diretório do código-fonte:

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

- 5.2. Adicione a segunda instrução de linha de impressão na página `index.php` para que mostre "A change is in the air!" e salve o arquivo. Adicione a alteração ao índice Git, faça commit dela e envie as alterações ao repositório Git remoto.

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

5.3. Acione uma build de aplicativo manualmente a partir do console da web.

Na barra de navegação, clique em Builds > Build Configs e, depois, clique no link **php-helloworld** para acessar a página **Build Config Details**. No menu Actions no canto superior direito da tela, clique em **Start Build**:

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, there's a navigation sidebar with options like Home, Operators, Workloads, Networking, Storage, Builds, Compute, and User Management. Under Builds, 'BuildConfigs' is selected. The main panel shows a 'BuildConfigs' list with one item: 'BC php-helloworld'. Below it, the 'BuildConfig details' page is displayed. This page includes sections for Details, YAML, Builds, Environment, and Events. The 'BuildConfig details' section shows the Name (php-helloworld), Type (Source), Namespace (youruser-console), Git repository (https://github.com/yourgithubuser/DO180-apps), Labels (app=php-helloworld, app.kubernetes.io/component=php-helloworld, etc.), Git ref (console), Context dir (php-helloworld), Build from (php:7.4-ubi8), Output to (php-helloworld:latest), Annotations (3 annotations), Run policy (Serial), and Triggers (Generic, GitHub, ImageChange, ConfigChange). To the right of the main content, there's a 'Actions' dropdown menu with options: Start build, Edit labels, Edit annotations, Edit BuildConfig, and Delete BuildConfig. The 'Start build' option is highlighted with a red box.

Figura 6.31: Inicie uma build de aplicativo

Você é redirecionado para uma página Build Details para a nova build. Clique na guia **Logs** para monitorar o progresso da build. A última linha de uma build bem-sucedida contém `Push successful`.

Quando a compilação for concluída, a implantação será iniciada. Acesse a seção **Workloads > Pods** e aguarde até que o novo pod seja implantado e esteja em execução.

- 5.4. Recarregue a URL `http://php-helloworld-${RHT_OCP4_DEV_USER}-console.${RHT_OCP4_WILDCARD_DOMAIN}` no navegador. A resposta do aplicativo corresponde ao código-fonte atualizado:



Figura 6.32: Saída de aplicativo web atualizada

6. Exclua o projeto. Na barra de navegação, clique em **Home > Projects**. Clique no ícone à direita da linha que contém uma entrada para o projeto `youruser-console`. Clique em **Delete Project** no menu que for exibido.

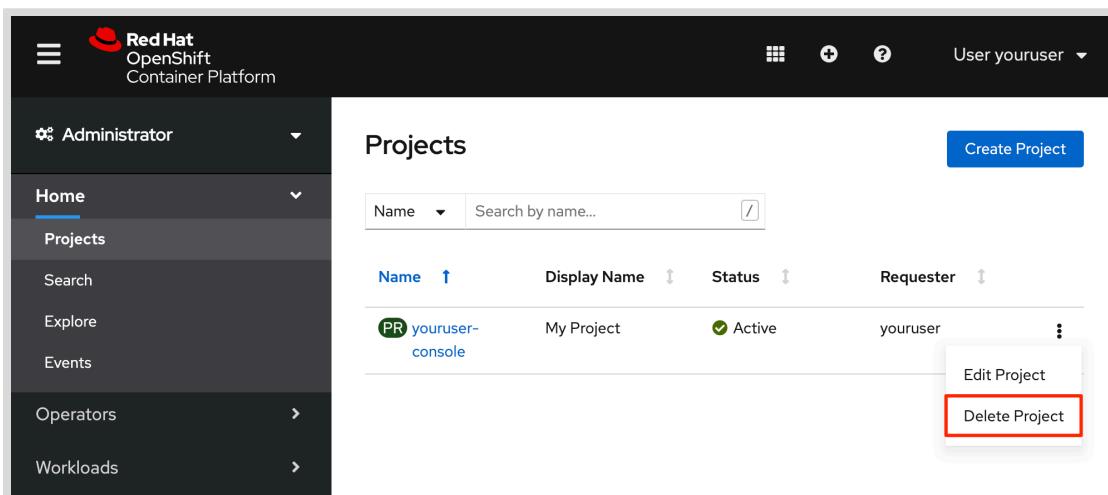


Figura 6.33: Excluir um projeto

Digite `youruser-console` na caixa de diálogo de confirmação e clique em **Delete**.

Encerramento

Na `workstation`, execute o script a seguir para concluir este laboratório.

```
[student@workstation php-helloworld]$ lab openshift-webconsole finish
```

Isso conclui o exercício orientado.

► Laboratório Aberto

Implantação de aplicativos conteinerizados no OpenShift

Resultados

Você deverá ser capaz de criar um aplicativo do OpenShift e acessá-lo através de um navegador da web.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Abra um terminal na **workstation** como usuário **student** e execute o seguinte comando:

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

Instruções

1. Carregue a configuração do seu ambiente de sala de aula. Faça login no cluster do OpenShift e crie um novo projeto para este exercício. Nomeie o projeto como \${RHT_OCP4_DEV_USER}-ocp.
2. Crie um aplicativo de conversor de temperatura chamado **temp**s em PHP usando a tag de fluxo de imagem **php:7.3**. O código-fonte está no repositório Git em <https://github.com/RedHatTraining/D0180-apps/> no diretório **temp**s. Você pode usar a interface de linha de comando do OpenShift ou o console da web para criar o aplicativo.
3. Verifique se é possível acessar o aplicativo com um navegador em [http://temp\\${RHT_OCP4_DEV_USER}-ocp.\\${RHT_OCP4_WILDCARD_DOMAIN}](http://temp${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN}).

Avaliação

Na **workstation**, execute o comando **lab openshift-review grade** para avaliar seu trabalho. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na **workstation**, execute o comando **lab openshift-review finish** para concluir este laboratório.

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

Isso结尾 the laboratório.

► Solução

Implantação de aplicativos conteinerizados no OpenShift

Resultados

Você deverá ser capaz de criar um aplicativo do OpenShift e acessá-lo através de um navegador da web.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula do Capítulo 1* antes de executar qualquer comando desta prática.

Abra um terminal na **workstation** como usuário **student** e execute o seguinte comando:

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

Instruções

1. Carregue a configuração do seu ambiente de sala de aula. Faça login no cluster do OpenShift e crie um novo projeto para este exercício. Nomeie o projeto como `${RHT_OCP4_DEV_USER} -ocp`.
 - 1.1. Carregue a configuração do ambiente de sala de aula.
Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- 1.2. Faça login no cluster do OpenShift.

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

- 1.3. Crie um novo projeto chamado " `${RHT_OCP4_DEV_USER} -ocp`" para os recursos que você criar durante este exercício:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-ocp
```

2. Crie um aplicativo de conversor de temperatura chamado `temps` em PHP usando a tag de fluxo de imagem `php:7.3`. O código-fonte está no repositório Git em <https://github.com/RedHatTraining/D0180-apps/> no diretório `temps`. Você pode usar a interface de linha de comando do OpenShift ou o console da web para criar o aplicativo.
 - 2.1. Se estiver usando a interface de linha de comando, execute os seguintes comandos:

```
[student@workstation ~]$ oc new-app \
> php:7.3~https://github.com/RedHatTraining/D0180-apps \
> --context-dir temps --name temps
--> Found image 688c0bd (2 months old) in image stream "openshift/php" under tag
"7.3" for "php:7.3"

Apache 2.4 with PHP 7.3
-----
PHP 7.3 available as container is a base platform ...output omitted...
...output omitted...

--> Creating resources ...
imagestream.image.openshift.io "temps" created
buildconfig.build.openshift.io "temps" created
deployment.apps "temps" created
service "temps" created
--> Success
Build scheduled, use 'oc logs -f bc/temps' to track its progress.
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose svc/temps'
Run 'oc status' to view your app.
```

2.2. Monitore o progresso da criação.

```
[student@workstation ~]$ oc logs -f bc/temps
Cloning "https://github.com/RedHatTraining/D0180-apps" ...
Commit: f7cd8963ef353d9173c3a21dcccf402f3616840b (Initial commit, including all
apps previously in course)
...output omitted...
Successfully pushed image-registry.openshift-image-registry.svc:5000/
${RHT_OCP4_DEV_USER}-temps
Push successful
```

2.3. Verifique se o aplicativo foi implantado.

```
[student@workstation ~]$ oc get pods -w
NAME        READY   STATUS    RESTARTS   AGE
temps-1-build   0/1     Completed   0          91s
temps-57d678bbdd-dlz9c   1/1     Running     0          58s
```

Pressione Ctrl+C para sair do comando `oc get pods -w`.

2.4. Exponha o serviço `temps` para criar uma rota externa para o aplicativo.

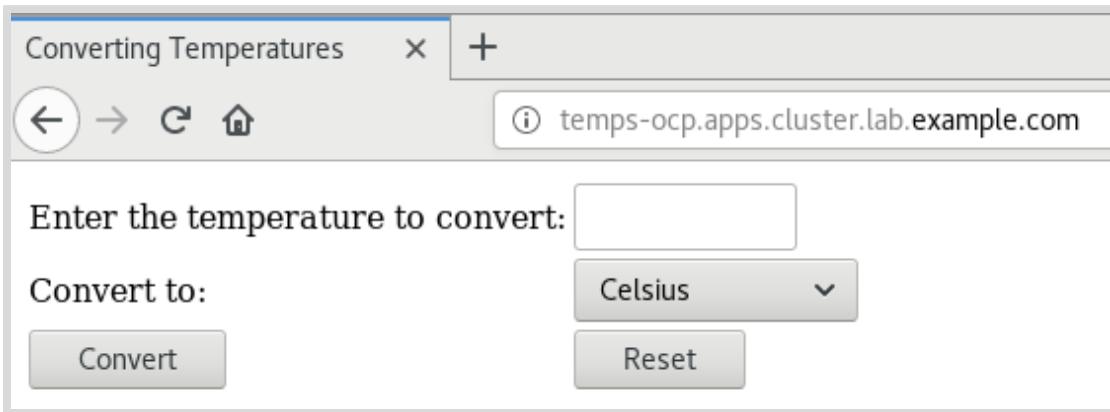
```
[student@workstation ~]$ oc expose svc/temps
route.route.openshift.io/temps exposed
```

- Verifique se é possível acessar o aplicativo com um navegador em `http://temps-${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN}`.

3.1. Determine a URL da rota.

```
[student@workstation ~]$ oc get route/temps
NAME      HOST/PORT
temps     temps-$${RHT_OCP4_DEV_USER} - ocp.$${RHT_OCP4_WILDCARD_DOMAIN} ...
```

- 3.2. Verifique se o aplicativo de conversor de temperatura funciona abrindo um navegador da web e navegando até a URL exibida na etapa anterior.



Avaliação

Na workstation, execute o comando `lab openshift-review grade` para avaliar seu trabalho. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na workstation, execute o comando `lab openshift-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

Sumário

Neste capítulo, você aprendeu que:

- O OpenShift Container Platform armazena definições de cada instância de recurso do OpenShift ou Kubernetes como um objeto no serviço de banco de dados distribuído do cluster, etcd. Tipos comuns de recursos são: Pod, Persistent Volume (PV), Persistent Volume Claim (PVC), Service (SVC), Route, Deployment, DeploymentConfig e Build Configuration (BC).
- Use o cliente oc de linha de comando do OpenShift para:
 - Criar, alterar e excluir projetos.
 - Criar recursos de aplicativos dentro de projetos.
 - Excluir, inspecionar, editar e exportar recursos dentro de um projeto.
 - Verificar logs de pods de aplicativos, implantações e operações de compilação.
- O comando oc new-app pode criar pods de aplicativos de muitas maneiras: a partir de uma imagem de contêiner existente hospedada em um registro de imagem, a partir de Containerfiles e de código-fonte bruto usando o processo Source-to-Image (S2I).
- Source-to-Image (S2I) é uma ferramenta que facilita a compilação de uma imagem de contêiner a partir de um código-fonte de aplicativo. Essa ferramenta recupera o código-fonte de um repositório Git, injeta o código-fonte em uma imagem de contêiner selecionada com base em uma linguagem ou tecnologia específica e produz uma nova imagem de contêiner que executa o aplicativo montado.
- Uma Route conecta um endereço IP voltado ao público externo e um nome de host DNS a um IP de serviço voltado ao público interno. Enquanto os serviços permitem acesso à rede entre os pods dentro de uma instância do OpenShift, as rotas dão permissão de acesso à rede aos pods de fora da instância do OpenShift.
- Você pode criar, compilar, implantar e monitorar aplicativos usando o console da web do OpenShift.

capítulo 7

Implantação de aplicativos com vários contêineres

Meta

Implantar aplicativos que são conteinerizados usando várias imagens de contêineres.

Objetivos

- Descrever as considerações para conteinerizar aplicativos com várias imagens de contêineres.
- Implantar um aplicativo de vários contêineres no OpenShift Platform.
- Implantar um aplicativo no OpenShift usando um template.

Seções

- Considerações sobre aplicativos de vários contêineres (e exercício orientado)
- Implantação de um aplicativo de vários contêineres no OpenShift (e exercício orientado)
- Implantação de um aplicativo de vários contêineres no OpenShift usando um template (e exercício orientado)

Laboratório

- Implantação de aplicativos com vários contêineres

Considerações sobre aplicativos de vários contêineres

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Descrever as considerações para conteinerizar aplicativos com várias imagens de contêineres.
- Aproveitar os conceitos de rede em contêineres.
- Criar um aplicativo de vários contêineres com o Podman.
- Descrever a arquitetura do aplicativo To Do List.

Aproveitar aplicativos com vários contêineres

Os exemplos mostrados até agora ao longo deste curso funcionaram bem com um único contêiner. Um aplicativo mais complexo, no entanto, pode obter os benefícios de implantar componentes diferentes em contêineres diferentes. Considere um aplicativo composto por um aplicativo web de front-end, um back-end do REST e um servidor de banco de dados. Esses componentes podem ter diferentes dependências, requisitos e ciclos de vida.

Embora seja possível orquestrar os contêineres de aplicativos de vários contêineres manualmente, o Kubernetes e o OpenShift fornecem ferramentas para facilitar a orquestração. A tentativa de gerenciar manualmente dezenas ou centenas de contêineres rapidamente se torna complicada. Nesta seção, voltaremos a usar o Podman para criar um aplicativo simples de vários contêineres para demonstrar as etapas manuais subjacentes para a orquestração de contêineres. Nas seções posteriores, você usará o Kubernetes e o OpenShift para orquestrar esses mesmos contêineres de aplicativos.

Descoberta de serviços em um aplicativo com vários contêineres

Contêineres rootfull com raiz

O Podman usa a Container Network Interface (CNI) para criar uma software-defined network (SDN) entre todos os contêineres no host. Salvo indicação em contrário, a CNI atribui um novo endereço IP a um contêiner quando ele é iniciado.

Cada contêiner expõe todas as portas aos outros contêineres na mesma SDN. Dessa maneira, os serviços são prontamente acessíveis na mesma rede. Os contêineres expõem portas a redes externas apenas por configuração explícita.

Devido à natureza dinâmica dos endereços IP de contêineres, os aplicativos não podem confiar em endereços IP fixos nem em nomes de hosts DNS fixos para se comunicar com serviços de middleware e outros serviços de aplicativos. Os contêineres com endereços IP dinâmicos podem se tornar um problema quando se trabalha com aplicativos com vários contêineres porque cada contêiner deve poder se comunicar com outros contêineres para usar os serviços dos quais ele depende.

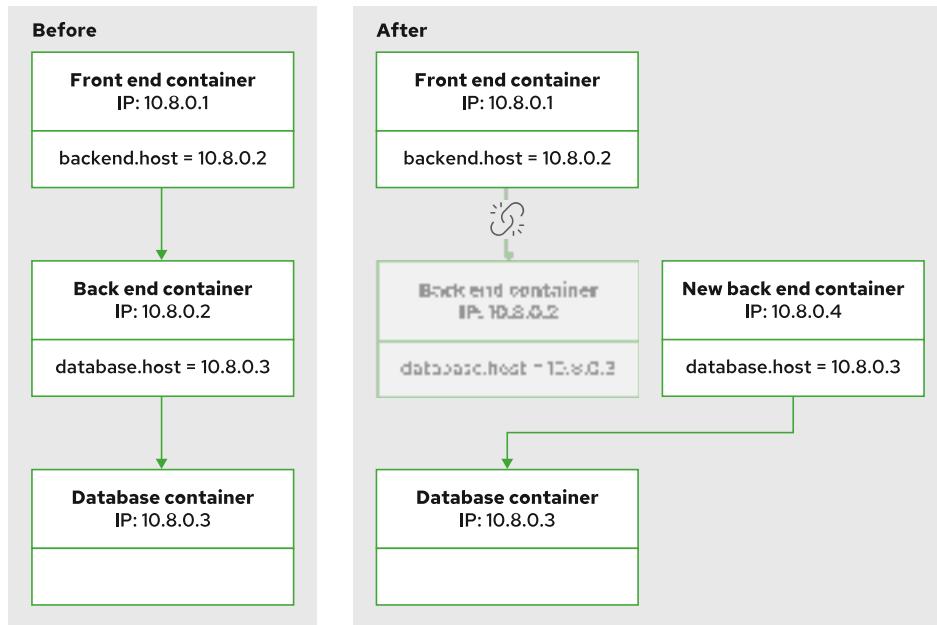


Figura 7.1: Uma reinicialização interrompe links de aplicativos de três camadas

Por exemplo, considere um aplicativo composto por um contêiner de front-end, um contêiner de back-end e um banco de dados. O contêiner de front-end precisa recuperar o endereço IP do contêiner de back-end. Da mesma forma, o contêiner de back-end precisa recuperar o endereço IP do contêiner do banco de dados. Além disso, o endereço IP seria alterado se um contêiner reiniciar. Por isso, é necessário um processo para garantir que alterações no IP açãoem uma atualização nos contêineres existentes.

O Kubernetes e o OpenShift oferecem possíveis soluções para o problema de detectabilidade de serviço e da natureza dinâmica da rede de contêineres. Algumas dessas soluções são abordadas mais adiante no capítulo.

Contêineres sem raiz

Contêineres sem raiz não são compatíveis com rede definida por software (SDN, Software-Defined Network). Portanto, o endereço IP do contêiner não está disponível para se comunicar com outros contêineres no host. Podemos alcançar uma rede entre contêineres sem raiz usando encaminhamento de porta. O encaminhamento de porta permite acesso externo a um serviço de contêiner a partir do host. O encaminhamento de porta foi discutido em Capítulo 3, Gerenciamento de contêineres.

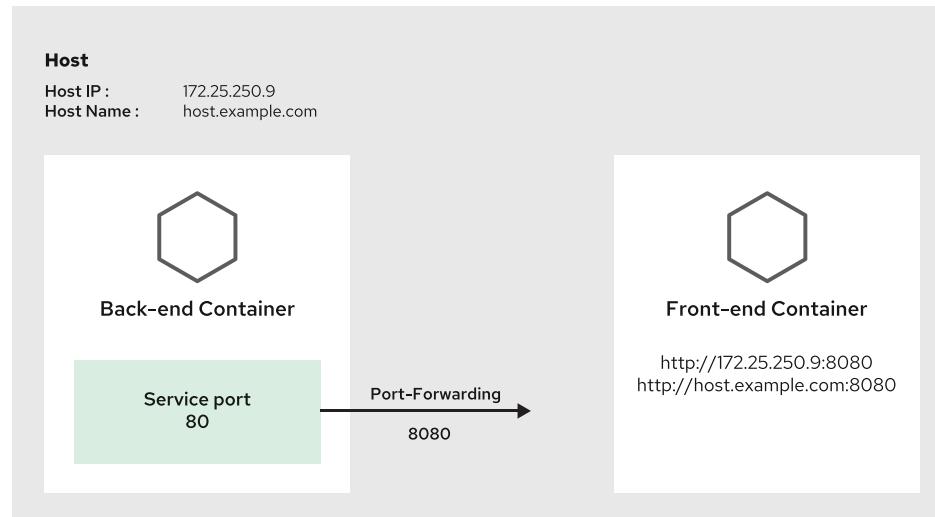


Figura 7.2: Aplicativos de vários contêineres para contêineres sem raiz

Por exemplo, considere um aplicativo composto por um contêiner de front-end e um de back-end. O contêiner de back-end tem um serviço em execução na porta específica 80. Esse serviço/porta não pode ser acessado de fora do contêiner. O contêiner de front-end precisa acessar o serviço/porta do contêiner de back-end. A primeira etapa é usar o *encaminhamento de porta* para o serviço necessário. Neste exemplo, encaminhamos a porta de serviço 80 para 8080. Agora, o contêiner de front-end pode acessar o serviço/porta do contêiner de back-end usando o endereço IP do host e a porta encaminhada.

Descrição do aplicativo To Do List

Muitos laboratórios neste curso fazem uso de um aplicativo To Do List. Esse aplicativo é dividido em três camadas, conforme ilustrado na seguinte imagem:

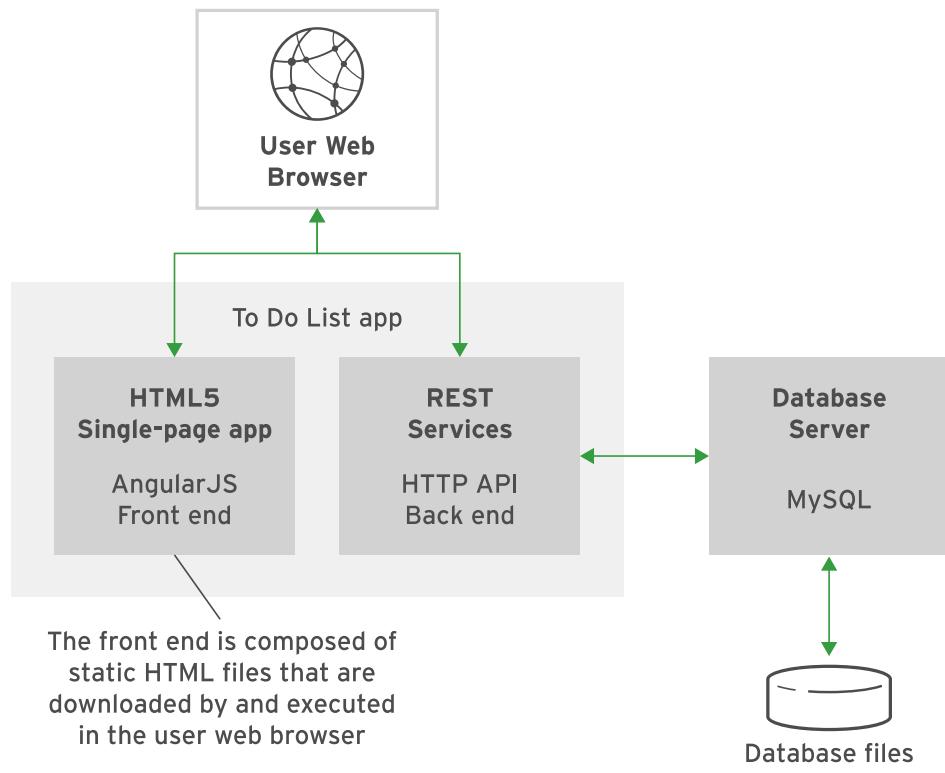


Figura 7.3: Arquitetura lógica do aplicativo To Do List

- A camada de apresentação é compilada como um front-end HTML5 de página única usando o AngularJS.
- A camada de negócios é composta por um back-end de API HTTP com o Node.js.
- A camada de persistência é baseada em um servidor de banco de dados MySQL.

A figura a seguir é uma captura de tela da interface web do aplicativo:

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

First Previous 1 Next Last

Add Task

Description:

Completed:

Clear **Save**

Figura 7.4: O aplicativo To Do List

À esquerda, temos uma tabela com itens a serem concluídos e, à direita, um formulário para adicionar um novo item.

O servidor de registros privado da sala de aula, `services.lab.example.com`, fornece o aplicativo em duas versões:

nodejs

Representa a maneira com que um desenvolvedor típico criaria o aplicativo como uma unidade única, sem o cuidado de dividi-la em camadas ou serviços.

nodejs_api

Mostra as alterações necessárias para dividir a apresentação do aplicativo e as camadas de negócios. Cada camada corresponde a uma imagem de contêiner isolada

As fontes de ambas as versões do aplicativo estão disponíveis na pasta `todoapp/nodejs` no repositório Git em: <https://github.com/RedHatTraining/D0180-apps.git>.

► Exercício Guiado

Implantação de aplicativo web e MySQL em contêineres Linux

Neste laboratório, você criará um script que executa e conecta via rede um contêiner de aplicativo do Node.js e o contêiner do MySQL.

Resultados

Você deverá ser capaz de conectar via rede contêineres para criar um aplicativo de várias camadas.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Você deve ter o código-fonte do aplicativo To Do List e os arquivos de laboratório na **workstation**. Para configurar o ambiente para o exercício, execute o seguinte comando:

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

Instruções

- 1. Faça login no Red Hat Container Catalog com sua conta da Red Hat. Se precisar se registrar na Red Hat, consulte as instruções em *Apêndice D, Criação de uma conta da Red Hat*.

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

- 2. Revise o Containerfile.

Usando o editor de sua preferência, abra e examine o Containerfile concluído localizado em /home/student/D0180/labs/multicontainer-design/deploy/nodejs/Containerfile.

- 3. Execute o comando ip addr para o endereço IP do host grep.

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

- 4. Explore as variáveis de ambiente.

Inspecione as variáveis de ambiente que permitem que o contêiner da API REST do Node.js se comunique com o contêiner do MySQL.

- 4.1. Veja o arquivo `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/nodejs-source/models/db.js` contendo a configuração do banco de dados fornecida abaixo:

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

- 4.2. Observe as variáveis de ambiente usadas pela API REST. Essas variáveis são expostas ao contêiner usando as opções `-e` com o comando `podman run` neste exercício orientado. Essas variáveis de ambiente são descritas abaixo.

MYSQL_DATABASE

O nome do banco de dados MySQL no contêiner `mysql`.

MYSQL_USER

O nome do usuário do banco de dados usado pelo contêiner `todoapi` para executar comandos MySQL.

MYSQL_PASSWORD

A senha do usuário do banco de dados que o contêiner `todoapi` usa para autenticar para o contêiner `mysql`.



nota

Os detalhes do host e da porta do contêiner MySQL são incorporados no aplicativo da API REST. O host, como mostrado acima no arquivo `db.js`, é o endereço IP do host, que é grep na etapa anterior.

- 5. Crie a imagem filho do aplicativo `To Do List` usando o `Containerfile` fornecido.

- 5.1. Crie a imagem filho.

Examine o script `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/build.sh` para ver como a imagem é criada. Execute os comandos a seguir para compilar a imagem filho.

```
[student@workstation nodejs]$ cd ~/D0180/labs/multicontainer-design/deploy/nodejs
[student@workstation nodejs]$ ./build.sh
Preparing build folder
Preparing build folder
STEP 1: FROM registry.redhat.io/rhel8/nodejs-12
Getting image source signatures
Copying blob 666be21779ed done
...output omitted...
Writing manifest to image destination
Storing signatures
STEP 2: ARG NEXUS_BASE_URL
```

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

**nota**

O script `build.sh` diminui as restrições de acesso de gravação ao diretório de compilação, permitindo que usuários não root instalem dependências.

Às vezes, a instabilidade da rede causa um erro `ERR!` com o script de construção. Para resolver esse problema, execute o script de com `build`.

- 5.2. Aguarde até que a compilação seja concluída e, então, execute o comando a seguir para verificar se a imagem foi criada com êxito:

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

- 6. Modifique o script existente para criar contêineres com as portas apropriadas, conforme definido na etapa anterior. Nesse script, a ordem dos comandos é determinada de modo a iniciar o contêiner `mysql` e, então, o contêiner `todoapi`, antes de conectar ao contêiner `mysql`. Depois de invocar todos os contêineres, há um tempo de espera de nove segundos, para que todos os contêineres tenham tempo de iniciar.
- 6.1. Edit o arquivo `run.sh` localizado em `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/networked` para inserir o comando `podman run` na linha apropriada para invocar o contêiner `mysql`. A tela a seguir mostra o comando `podman` exato que deve ser inserido no arquivo.

```
podman run -d --name mysql -e MYSQL_DATABASE=items -e MYSQL_USER=user1 \
-e MYSQL_PASSWORD=mypa55 -e MYSQL_ROOT_PASSWORD=r00tpa55 \
-v $PWD/work/data:/var/lib/mysql/data \
-p 30306:3306 \
registry.redhat.io/rhel8/mysql-80:1
```

No comando anterior, `MYSQL_DATABASE`, `MYSQL_USER` e `MYSQL_PASSWORD` são preenchidos com as credenciais para acessar o banco de dados do MySQL. Essas variáveis de ambiente são necessárias para que o contêiner `mysql` seja executado. Além disso, a pasta local `$PWD/work/data` é montada como volume no sistema de arquivos do contêiner.

- 6.2. No mesmo arquivo `run.sh`, insira outro comando `podman run` na linha apropriada para executar o contêiner `todoapi`. A tela a seguir mostra o comando `docker` que deve ser inserido no arquivo.

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

**nota**

Depois de inserir cada comando `podman run` no script `run.sh`, certifique-se de que também haja um comando `sleep 9`. Se você precisar repetir essa etapa, será necessário excluir o diretório `work` e seu conteúdo antes de executar novamente o script `run.sh`.

- 6.3. Verifique se o script `run.sh` corresponde ao script de solução localizado em `/home/student/D0180/solutions/multicontainer-design/deploy/nodejs/networked/run.sh`.
 - 6.4. Salve o arquivo e saia do editor.
- 7. Execute os contêineres.
- 7.1. Use o comando a seguir para executar o script que você atualizou para executar os contêineres `mysql` e `todoapi`.
- ```
[student@workstation nodejs]$ cd \
> /home/student/D0180/labs/multicontainer-design/deploy/nodejs/networked
[student@workstation networked]$./run.sh
```
- 7.2. Verifique se os contêineres foram iniciados com êxito.
- ```
[student@workstation networked]$ podman ps \
> --format="table {{.ID}} {{.Names}} {{.Image}} {{.Status}}"
ID          Names    Image                               Status
c74b4709e3ae  todoapi  localhost/do180/todonodejs:latest  Up 3 minutes ago
3bc19f74254c  mysql    registry.redhat.io/rhel8/mysql-80:1  Up 3 minutes ago
```

- 8. Preencha o banco de dados `items` com a tabela `Projects`.

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

- 9. Examine as variáveis de ambiente do contêiner da API.

Execute o comando a seguir para explorar as variáveis de ambiente expostas no contêiner da API.

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

► 10. Teste o aplicativo.

- 10.1. Execute um comando `curl` para testar a API REST para o aplicativo `To Do List`.

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

A opção `-w "\n"` com o comando `curl` permite que o prompt do shell seja exibido ao lado da linha em vez de mesclar a saída na mesma linha.

- 10.2. Abra o Firefox na `workstation` e acesse `http://127.0.0.1:30080/todo/`. Você deverá ver o aplicativo `To Do List`.



nota

Certifique-se de acrescentar a barra à direita (/).

- 10.3. Altere para o diretório `/home/student`.

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

Encerramento

Na `workstation`, execute o script `lab multicontainer-design finish` para concluir esse exercício.

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

Isso conclui o exercício orientado.

Implantação de um aplicativo de vários contêineres no OpenShift

Objetivos

Depois de concluir esta seção, os alunos deverão ser capazes de:

- Descrever as diferenças entre o Podman e o Kubernetes
- Implantar um aplicativo de vários contêineres no OpenShift.

Comparação entre o Podman e o Kubernetes

Usar variáveis de ambiente permite que você compartilhe informações entre contêineres com o Podman. Entretanto, ainda haverá limitações e será necessário algum trabalho manual para assegurar que todas as variáveis de ambiente permaneçam em sincronia, especialmente ao trabalhar com muitos contêineres. O Kubernetes oferece uma abordagem para resolver esse problema criando serviços para seus contêineres, conforme visto em capítulos anteriores.

Serviços no Kubernetes

Os pods são anexados a um namespace do Kubernetes, que o OpenShift chama de *projeto*. Quando um pod é iniciado, o Kubernetes adiciona automaticamente um conjunto de variáveis de ambiente para cada serviço definido no mesmo namespace.

Qualquer serviço no Kubernetes gera variáveis de ambiente para o endereço IP e o número da porta onde o serviço está disponível. O Kubernetes injeta automaticamente essas variáveis de ambiente nos contêineres de pods no mesmo namespace. Essas variáveis de ambiente normalmente seguem uma convenção:

- *Letra maiúscula*: todas as variáveis de ambiente são definidas usando nomes em letra maiúscula.
- *Snakecase*: qualquer variável de ambiente criada por um serviço é geralmente composta por várias palavras separadas por um sublinhado (_).
- *Primeiro o nome do serviço*: a primeira palavra de uma variável de ambiente criada por um serviço é o nome do serviço.
- *Tipo de protocolo*: a maioria das variáveis de ambiente de rede inclui o tipo de protocolo (TCP ou UDP).

Essas são as variáveis de ambiente geradas pelo Kubernetes para um serviço:

- <SERVICE_NAME>_SERVICE_HOST: representa o endereço IP ativado por um serviço para acessar um pod.
- <SERVICE_NAME>_SERVICE_PORT: representa a porta em que a porta do servidor é listada.
- <SERVICE_NAME>_PORT: representa o endereço, a porta e o protocolo oferecidos pelo serviço para acesso externo.
- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>: define um alias para <SERVICE_NAME>_PORT.

- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PROTO: identifica o tipo de protocolo (TCP ou UDP).
- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PORT: define um alias para <SERVICE_NAME>_SERVICE_PORT.
- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_ADDR: define um alias para <SERVICE_NAME>_SERVICE_HOST.

Essas são as variáveis de ambiente geradas pelo Kubernetes para um serviço.

Por exemplo, dado o seguinte serviço:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
  name: mysql
spec:
  ports:
    - protocol: TCP
      - port: 3306
  selector:
    name: mysql
```

As seguintes variáveis de ambiente estão disponíveis para cada pod criado após o serviço, no mesmo namespace:

```
MYSQL_SERVICE_HOST=10.0.0.11
MYSQL_SERVICE_PORT=3306
MYSQL_PORT=tcp://10.0.0.11:3306
MYSQL_PORT_3306_TCP=tcp://10.0.0.11:3306
MYSQL_PORT_3306_TCP_PROTO=tcp
MYSQL_PORT_3306_TCP_PORT=3306
MYSQL_PORT_3306_TCP_ADDR=10.0.0.11
```



nota

Outros nomes de variáveis de ambiente <SERVICE_NAME>_PORT_* relevantes são definidos com base no protocolo. O endereço IP e o número da porta estão definidos na variável de ambiente <SERVICE_NAME>_PORT. Por exemplo, a entrada MYSQL_PORT=tcp://10.0.0.11:3306 leva à criação de variáveis de ambiente com nomes como MYSQL_PORT_3306_TCP, MYSQL_PORT_3306_TCP_PROTO, MYSQL_PORT_3306_TCP_PORT e MYSQL_PORT_3306_TCP_ADDR. Se o componente do protocolo de uma variável de ambiente for indefinido, o Kubernetes usará o protocolo TCP e atribuirá os nomes das variáveis de acordo adequadamente.

► Exercício Guiado

Criação de um aplicativo no OpenShift.

Neste exercício, você implantará o aplicativo To Do List no OpenShift Container Platform.

Resultados

Você deverá ser capaz de criar e implantar um aplicativo no OpenShift Container Platform.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Você deve ter o código-fonte do aplicativo To Do List e os arquivos de laboratório na workstation. Para fazer download dos arquivos do laboratório e verificar o status do cluster do OpenShift, execute o seguinte comando em uma nova janela de terminal.

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

Instruções

- 1. Crie o aplicativo To Do List a partir do arquivo YAML fornecido.

- 1.1. Faça login na OpenShift Container Platform.

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.

...output omitted...

Using project "default".
```

Se o comando `oc login` perguntar sobre o uso de conexões não seguras, responda `y` (sim).

- 1.2. Crie um novo projeto *application* no OpenShift para usar neste exercício. Execute o comando a seguir para criar o projeto *application*.

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

- 1.3. Revise o arquivo YAML.

Usando o editor de sua preferência, abra e examine o arquivo de aplicativo localizado em `/home/student/D0180/labs/multicontainer-application/todo-`

`app.yml`. Observe os recursos a seguir definidos no `todo-app.yml` e revise suas configurações.

- A definição de pod `todoapi` define o aplicativo do Node.js.
- A definição de pod `mysql` define o banco de dados MySQL.
- O serviço `todoapi` fornece conectividade ao pod do aplicativo do Node.js
- O serviço `mysql` fornece conectividade ao pod do banco de dados MySQL.
- A definição da solicitação de volume persistente `dbclaim` define o volume `/var/lib/mysql/data` do MySQL.

1.4. Crie recursos do aplicativo com determinado arquivo yaml.

Use o comando `oc create` para criar os recursos do aplicativo. Execute o seguinte comando na janela de terminal:

```
[student@workstation ~]$ cd /home/student/D0180/labs/multicontainer-application
[student@workstation multicontainer-application]$ oc create -f todo-app.yml
pod/mysql created
pod/todoapi created
service/todoapi created
service/mysql created
persistentvolumeclaim/dbclaim created
```

1.5. Revise a implantação.

Revise o status da implantação usando o comando `oc get pods` com a opção `-w` para continuar a monitorar o status do pod. Aguarde até que ambos os contêineres estejam em execução. Pode levar algum tempo para que ambos os pods sejam iniciados.

```
[student@workstation multicontainer-application]$ oc get pods -w
NAME      READY    STATUS      RESTARTS   AGE
todoapi   1/1     Running    0          27s
mysql     1/1     Running    0          27s
```

Pressione `Ctrl+C` para sair do comando.

► 2. Conecte-se ao servidor de banco de dados MySQL e preencha os dados para o banco de dados `item`.

2.1. Na máquina `workstation`, configure o encaminhamento de porta entre a `workstation` e o pod de banco de dados em execução no OpenShift usando a porta 3306. O terminal travará depois de executar o comando.

```
[student@workstation multicontainer-application]$ oc port-forward mysql 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
```

2.2. Na máquina `workstation`, abra outro terminal e preencha os dados do servidor MySQL usando o cliente MySQL.

```
[student@workstation ~]$ cd /home/student/D0180/labs/multicontainer-application
[student@workstation multicontainer-application]$ mysql -uuser1 \
> -h 127.0.0.1 -pmypa55 -P3306 items < db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- 2.3. Feche o terminal e retorne ao anterior. Conclua o processo de encaminhamento de porta pressionando **Ctrl+C**.

```
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::]:3306 -> 3306
Handling connection for 3306
^C
```

► 3. Exponha o serviço.

Para permitir que o aplicativo `To Do List` fique acessível através do roteador OpenShift e fique disponível como um FQDN público, use o comando `oc expose` para expor o serviço `todoapi`.

Execute o comando a seguir na janela de terminal.

```
[student@workstation multicontainer-application]$ oc expose service todoapi
route.route.openshift.io/todoapi exposed
```

► 4. Teste o aplicativo.

- 4.1. Encontre o FQDN do aplicativo executando o comando `oc status` e observe o FQDN do aplicativo.

Execute o comando a seguir na janela de terminal.

```
[student@workstation multicontainer-application]$ oc status | grep -o "http:.com"
http://todoapi-${RHT_OCP4_DEV_USER}-application.${RHT_OCP4_WILDCARD_DOMAIN}
```

- 4.2. Use `curl` para testar a API do REST para o aplicativo `To Do List`.

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

A opção `-w "\n"` com o comando `curl` permite que o prompt do shell seja exibido ao lado da linha em vez de mesclar a saída na mesma linha.

- 4.3. Altere para o diretório `/home/student`.

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

- 4.4. Abra o Firefox na `workstation` e aponte seu navegador para `http://todoapi-${RHT_OCP4_DEV_USER}-application.${RHT_OCP4_WILDCARD_DOMAIN}/todo/`; você deverá ver o aplicativo `To Do List`.

**nota**

A barra à direita na URL mencionada acima é necessária. Se você não incluir essa barra no URL, poderão ocorrer problemas com o aplicativo.

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

First Previous **1** Next Last

Add Task

Description:

Completed:

Clear **Save**

Figura 7.5: Aplicativo To Do List

Encerramento

Na workstation, execute o script `lab multicontainer-application finish` para concluir esse laboratório.

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

Isso conclui o exercício orientado.

Implantação de um aplicativo de vários contêineres no OpenShift usando um template

Depois de concluir esta seção, os alunos deverão ser capazes de implantar um aplicativo de vários contêineres no OpenShift usando um template.

Análise do esqueleto de um template

Implantar um aplicativo no OpenShift Container Platform geralmente exige a criação de vários recursos relacionados dentro de um projeto. Por exemplo, um aplicativo web pode exigir que um recurso `BuildConfig`, `Deployment`, `Service` e `Route` seja executado em um projeto do OpenShift. Geralmente, os atributos desses recursos têm o mesmo valor, como o atributo `name` de um recurso.

Os modelos do OpenShift fornecem uma maneira de simplificar a criação de recursos que um aplicativo exige. Um modelo define um conjunto de recursos relacionados que devem ser criados juntas, bem como um conjunto de parâmetros de aplicativos. Os atributos dos recursos do modelo são geralmente definidos em termos dos parâmetros do modelo, como o atributo `name` de um recurso.

Por exemplo, um aplicativo pode consistir de um aplicativo web de front-end e um servidor de banco de dados. Cada um é composto por um recurso de serviço e um recurso de implantação. Eles compartilham um conjunto de credenciais (parâmetros) para o front-end autenticar no back-end. O template pode ser processado especificando os parâmetros ou permitindo que eles sejam automaticamente gerados (por exemplo, para uma senha de banco de dados exclusiva) para instanciar a lista de recursos no template como um aplicativo coeso.

O instalador do OpenShift cria vários templates por padrão no namespace `openshift`. Execute o comando `oc get templates` com a opção `-n openshift` para listar esses templates pré-instalados:

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

A definição YAML de um template pode ser modificada para atender às necessidades de seus aplicativos. Você fará isso em um laboratório futuro.

Veja abaixo uma definição de template YAML:

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

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

- ➊ Define uma lista de tags arbitrárias para associar a esse modelo. Insira qualquer uma dessas tags na interface do usuário para encontrar esse modelo.
- ➋ Define o nome do template.
- ➌ A seção `objects` define a lista de recursos do OpenShift para este template. Esse template cria quatro recursos: um `Secret`, um `Service`, um `PersistentVolumeClaim` e um `Deployment`.
- ➍ Todos os quatro objetos de recursos têm seus nomes definidos do valor do parâmetro `DATABASE_SERVICE_NAME`.
- ➎ A seção `parameters` contém uma lista de nove parâmetros.
- ➏ Os recursos do template geralmente definem seus atributos usando os valores desses parâmetros, conforme demonstrado com o parâmetro `DATABASE_SERVICE_NAME`.
- ➐ Se você não especificar um valor para o parâmetro `MYSQL_PASSWORD` ao criar um aplicativo com esse template, o OpenShift gera uma senha que corresponde a essa expressão regular.

Você pode publicar um novo template no cluster do OpenShift para que outros desenvolvedores possam compilar aplicativos a partir desse template.

Suponha que você tenha um aplicativo de lista de tarefas chamado `todo` que exige um objeto `Deployment`, `Service` e `Route` do OpenShift para implantação. Você cria um arquivo de definição de template YAML que define atributos para esses recursos do OpenShift, juntamente com definições para qualquer parâmetro necessário. Pressupondo que o template seja definido no arquivo `todo-template.yaml`, use o comando `oc create` para publicar o template do aplicativo:

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

Por padrão, o template é criado sob o projeto atual, a menos que você especifique um projeto diferente usando a opção `-n`, conforme mostrado no seguinte exemplo:

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



Importante

Qualquer template criado no namespace `openshift` (projeto do OpenShift) fica disponível no console da web em `Developer Catalog` da perspectiva `Developer`. Além disso, qualquer template criado no projeto atual será acessível a partir desse projeto.

Parâmetros

Os modelos definem um conjunto de parâmetros, que são valores atribuídos. Os recursos do OpenShift definidos no modelo podem obter valores de configuração ao se referirem aos *parâmetros nomeados*. Os parâmetros em um template podem ter valores padrão, mas eles são opcionais. Quaisquer valores padrão podem ser substituídos ao processar o template.

Cada valor de parâmetro pode ser definido explicitamente usando o comando `oc process` ou gerado pelo OpenShift de acordo com a configuração de parâmetros.

Há duas maneiras de listar os parâmetros disponíveis a partir de um template. A primeira é usando o comando `oc describe`:

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

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

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

...output omitted...

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

Object Labels: template=mysql-persistent-template

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

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

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

Objects:

```
Secret      ${DATABASE_SERVICE_NAME}
Service     ${DATABASE_SERVICE_NAME}
PersistentVolumeClaim ${DATABASE_SERVICE_NAME}
Deployment   ${DATABASE_SERVICE_NAME}
```

A segunda maneira é usando o `oc process` com a opção `--parameters`:

```
[user@host ~]$ oc process --parameters mysql-persistent -n openshift
NAME          DESCRIPTION      GENERATOR      VALUE
MEMORY_LIMIT  Maximum a...    expression      512Mi
NAMESPACE     The OpenS...    expression      openshift
DATABASE_SERVICE_NAME The name ...    expression      mysql
MYSQL_USER    Username ...   expression      user[A-Z0-9]{3}
MYSQL_PASSWORD Password ...  expression      [a-zA-Z0-9]{16}
MYSQL_ROOT_PASSWORD Password ... expression      [a-zA-Z0-9]{16}
MYSQL_DATABASE Name of t...  expression      sampledb
VOLUME_CAPACITY Volume sp...  expression      1Gi
MYSQL_VERSION Version o...  expression      8.0
```

Processamento de um template usando a CLI

Ao processar um template, você gera uma lista de recursos para criar um novo aplicativo. Para processar um template, use o comando `oc process`:

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

O comando anterior processa um arquivo de template no formato JSON ou YAML e retorna a lista de recursos para a saída padrão. O formato da lista de recursos de saída é JSON. Para produzir a lista de recursos no formato YAML, use o `-o yaml` com o comando `oc process`:

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

Outra opção é processar um template a partir do projeto atual ou do projeto `openshift`:

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



nota

O comando `oc process` retorna uma lista de recursos para a saída padrão. Essa saída pode ser redirecionada a um arquivo:

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

Os templates geralmente geram recursos com atributos configuráveis baseados nos parâmetros do template. Para substituir um parâmetro, use a opção `-p`, seguida de um par `<name>=<value>`.

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

Para criar o aplicativo, use o arquivo de definição de recurso YAML gerado:

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

Como alternativa, é possível processar o template e criar o aplicativo sem salvar um arquivo de definição de recurso usando um pipe do UNIX:

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

Para usar um modelo no projeto openshift a fim de criar um aplicativo em seu projeto, primeiro exporte o modelo:

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

Em seguida, identifique os valores apropriados para os parâmetros do template e processe o template:

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

Você também pode usar duas barras (//) para fornecer o namespace como parte do nome do template:

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

Como alternativa, é possível criar um aplicativo usando o comando `oc new-app`, passando o nome do template como argumento da opção `--template`:

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



Referências

As informações de desenvolvedor sobre templates podem ser encontradas na seção *Using Templates* da documentação do OpenShift Container Platform:

Guia do desenvolvedor

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/images/index#using-templates

► Exercício Guiado

Criação de um aplicativo com um template

Neste exercício, você implantará o aplicativo To Do List no OpenShift Container Platform usando um template para definir os recursos necessários para executar seu aplicativo.

Resultados

Você deverá ser capaz de compilar e implantar um aplicativo no OpenShift Container Platform usando um template JSON fornecido.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Você deve ter o código-fonte do aplicativo To Do List e os arquivos de laboratório na **workstation**. Para fazer download dos arquivos do laboratório e verificar o status do cluster do OpenShift, execute o seguinte comando em uma nova janela de terminal.

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

Instruções

- 1. Crie o aplicativo To Do List a partir do template JSON fornecido.

- 1.1. Carregue a configuração do ambiente de sala de aula.

Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- 1.2. Faça login na OpenShift Container Platform.

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

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

```
Using project "default".
```

Se o comando `oc login` perguntar sobre o uso de conexões não seguras, responda `y` (sim).

- 1.3. Crie um novo projeto *template* no OpenShift para usar neste exercício. Execute o comando a seguir para criar o projeto *template*.

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

1.4. Revise o template.

Usando o editor de sua preferência, abra e examine o template localizado em `/home/student/D0180/labs/multicontainer-openshift/todo-template.json`. Observe os recursos a seguir definidos no template e revise suas configurações.

- A definição de pod `todoapi` define o aplicativo do Node.js.
- A definição de pod `mysql` define o banco de dados MySQL.
- O serviço `todoapi` fornece conectividade ao pod do aplicativo do Node.js
- O serviço `mysql` fornece conectividade ao pod do banco de dados MySQL.
- A definição da solicitação de volume persistente `dbclaim` define o volume `/var/lib/mysql/data` do MySQL.

1.5. Processe o template e crie os recursos do aplicativo.

Use o comando `oc process` para processar o arquivo de template. Esse template exige que o namespace Quay.io recupere as imagens de contêiner. Use o comando `pipe` para enviar o resultado para o comando `oc create`.

Execute o seguinte comando na janela de terminal:

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

1.6. Revise a implantação.

Revise o status da implantação usando o comando `oc get pods` com a opção `-w` para continuar a monitorar o status do pod. Aguarde até que ambos os contêineres estejam em execução. Pode levar algum tempo para que ambos os pods sejam iniciados.

```
[student@workstation multicontainer-openshift]$ oc get pods -w
NAME      READY    STATUS          RESTARTS   AGE
mysql     0/1     ContainerCreating   0          27s
todoapi   1/1     Running         0          27s
mysql     1/1     Running         0          27s
```

Pressione `Ctrl+C` para sair do comando.

- 2. Conecte-se ao servidor de banco de dados MySQL e preencha os dados para o banco de dados `item`.

capítulo 7 | Implantação de aplicativos com vários contêineres

- 2.1. Na máquina workstation, configure o encaminhamento de porta entre a workstation e o pod de banco de dados em execução no OpenShift usando a porta 3306. O terminal travará depois de executar o comando.

```
[student@workstation multicontainer-openshift]$ oc port-forward mysql 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
```

- 2.2. Na máquina workstation, abra outro terminal e preencha os dados do servidor MySQL usando o cliente MySQL.

```
[student@workstation ~]$ cd /home/student/D0180/labs/multicontainer-openshift
[student@workstation multicontainer-openshift]$ mysql -uuser1 \
> -h 127.0.0.1 -pmypa55 -P3306 items < db.sql
mysql: [Warning] Using a password on the command line interface can be insecure.
```

- 2.3. Feche o terminal e retorne ao anterior. Conclua o processo de encaminhamento de porta pressionando **Ctrl+C**.

```
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
Handling connection for 3306
^C
```

► 3. Exponha o serviço.

Para permitir que o aplicativo `To Do List` fique acessível através do roteador OpenShift e fique disponível como um FQDN público, use o comando `oc expose` para expor o serviço `todoapi`.

Execute o comando a seguir na janela de terminal.

```
[student@workstation multicontainer-openshift]$ oc expose service todoapi
route.route.openshift.io/todoapi exposed
```

► 4. Teste o aplicativo.

- 4.1. Encontre o FQDN do aplicativo executando o comando `oc status` e observe o FQDN do aplicativo.

Execute o comando a seguir na janela de terminal.

```
[student@workstation multicontainer-openshift]$ oc status | grep -o "http:.*com"
http://todoapi-${{RHT_OCP4_DEV_USER}}-template.${{RHT_OCP4_WILDCARD_DOMAIN}}
```

- 4.2. Use `curl` para testar a API do REST para o aplicativo `To Do List`.

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

A opção `-w "\n"` com o comando `curl` permite que o prompt do shell seja exibido ao lado da linha em vez de mesclar a saída na mesma linha.

4.3. Altere para o diretório /home/student.

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

4.4. Abra o Firefox na workstation e aponte seu navegador para `http://todoapi-
${RHT_OCP4_DEV_USER}-template.${RHT_OCP4_WILDCARD_DOMAIN}/
todo/`; você deverá ver o aplicativo To Do List.



nota

A barra à direita na URL mencionada acima é necessária. Se você não incluir essa barra no URL, poderão ocorrer problemas com o aplicativo.

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

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

On the right, there is a 'Add Task' form with fields for 'Description' (containing 'Add Description.') and 'Completed' (with an unchecked checkbox). Below the form are 'Clear' and 'Save' buttons. At the bottom of the page, there is a navigation bar with buttons for 'First', 'Previous', '1', 'Next', and 'Last'.

Figura 7.6: Aplicativo To Do List

Encerramento

Na workstation, execute o script a seguir para concluir este laboratório.

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

Isso conclui o exercício orientado.

► Laboratório Aberto

Implantação de aplicativos com vários contêineres

Resultados

Você deverá ser capaz de criar um aplicativo do OpenShift composto por vários contêineres e acessá-lo em um navegador.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula do Capítulo 1* antes de executar qualquer comando desta prática.

Abra um terminal na `workstation` como usuário `student` e execute os seguintes comandos:

```
[student@workstation ~]$ lab multicontainer-review start  
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-review
```

Instruções

1. Faça login no cluster do OpenShift e crie um novo projeto para este exercício. Nomeie o projeto como `${RHT_OCP4_DEV_USER}-deploy`.
2. Compile a imagem de contêiner do banco de dados localizado no diretório `images/mysql`, marque-a com `do180-mysql-80-rhel8` e publique-a no repositório Quay.io.
3. Compile a imagem de contêiner PHP localizado no `images/quote-php`, marque-a com `do180-quote-php` e publique-a no repositório Quay.io.



Cuidado

Certifique-se de que os dois repositórios sejam públicos em `quay.io` para que o OpenShift possa obter as imagens dele. Consulte a seção `Repositories Visibility` do Apêndice C, *Criação de uma conta no Quay* para ler detalhes sobre como alterar a visibilidade do repositório.

4. Acesse o diretório `/home/student/D0180/labs/multicontainer-review/` e revise o arquivo de template `quote-php-template.json` fornecido.
5. Faça upload do template de aplicativo PHP para que qualquer desenvolvedor com acesso ao seu projeto possa usá-lo.
6. Processe o template carregado, crie os recursos de aplicativo e confirme o status.
7. Exponha o serviço `quote-php`.
Permita que o aplicativo PHP `Quote` fique acessível através do roteador do OpenShift e a partir de uma rede externa.
8. Teste o aplicativo com `curl` e verifique se ele envia uma mensagem inspiradora.

Avaliação

Avalie seu trabalho executando o comando `lab multicontainer-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `lab multicontainer-review finish`, execute o comando `workstation` para concluir este laboratório.

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

Isso conclui o laboratório.

► Solução

Implantação de aplicativos com vários contêineres

Resultados

Você deverá ser capaz de criar um aplicativo do OpenShift composto por vários contêineres e acessá-lo em um navegador.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula do Capítulo 1* antes de executar qualquer comando desta prática.

Abra um terminal na `workstation` como usuário `student` e execute os seguintes comandos:

```
[student@workstation ~]$ lab multicontainer-review start
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-review
```

Instruções

1. Faça login no cluster do OpenShift e crie um novo projeto para este exercício. Nomeie o projeto como `${RHT_OCP4_DEV_USER}-deploy`.
 - 1.1. Na máquina `workstation`, faça login como o usuário fornecido no primeiro exercício.

```
[student@workstation multicontainer-review]$ source /usr/local/etc/ocp4.config
[student@workstation multicontainer-review]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
```

You don't have any projects. You can try to create a new project, by running

```
oc new-project <projectname>
```

Se o comando `oc login` perguntar sobre o uso de conexões não seguras, responda `y` (sim).

- 1.2. Crie um novo projeto no OpenShift chamado `deploy` e prefixado pelo seu nome de usuário OpenShift:

```
[student@workstation multicontainer-review]$ oc new-project \
> ${RHT_OCP4_DEV_USER}-deploy
Now using project ...output omitted...
```

2. Compile a imagem de contêiner do banco de dados localizado no diretório `images/mysql`, marque-a com `d0180-mysql-80-rhel8` e publique-a no repositório Quay.io.

- 2.1. Faça login no Red Hat Container Catalog com sua conta da Red Hat.

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

- 2.2. Compile a imagem do banco de dados do MySQL usando o Dockerfile fornecido no diretório `images/mysql`.

```
[student@workstation multicontainer-review]$ cd images/mysql
[student@workstation mysql]$ podman build -t do180-mysql-80-rhel8 .
STEP 1: FROM registry.redhat.io/rhel8/mysql-80:1
...output omitted...
STEP 4: COMMIT do180-mysql-80-rhel8
397a...5cfb
```

- 2.3. Envie a imagem do MySQL para o repositório Quay.io.

Para tornar a imagem disponível para o OpenShift usar no template, atribua a ela a tag `quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-80-rhel8` e envie-a ao registro `quay.io`. Para enviar imagens para `quay.io` você primeiro, é necessário fazer login com suas próprias credenciais.

```
[student@workstation mysql]$ podman login quay.io -u ${RHT_OCP4_QUAY_USER}
Password: your_quay_password
Login Succeeded!
```

Para marcar e enviar a imagem, execute os comandos a seguir na janela de terminal.

```
[student@workstation mysql]$ podman tag do180-mysql-80-rhel8 \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-80-rhel8
[student@workstation mysql]$ podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-80-rhel8
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

Retorne ao diretório anterior.

```
[student@workstation mysql]$ cd ~/DO180/labs/multicontainer-review
```

3. Compile a imagem de contêiner PHP localizado no `images/quote-php`, marque-a com `do180-quote-php` e publique-a no repositório Quay.io.



Cuidado

Certifique-se de que os dois repositórios sejam públicos em `quay.io` para que o OpenShift possa obter as imagens dele. Consulte a seção `Repositories Visibility` do ??? para ler detalhes sobre como alterar a visibilidade do repositório.

- 3.1. Compile a imagem PHP usando o Containerfile fornecido no diretório `images/quote-php`.

```
[student@workstation multicontainer-review]$ cd images/quote-php
[student@workstation quote-php]$ podman build -t do180-quote-php .
STEP 1: FROM registry.access.redhat.com/ubi8/ubi
...output omitted...
STEP 8: COMMIT do180-quote-php
271f...525d
```

- 3.2. Marque e envie a imagem PHP ao registro Quay.io.

Para tornar a imagem disponível para o OpenShift usar no template, atribua a ela a tag `quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php` e envie-a ao Quay.io.

```
[student@workstation quote-php]$ podman tag do180-quote-php \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php
[student@workstation quote-php]$ podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

4. Acesse o diretório `/home/student/D0180/labs/multicontainer-review/` e revise o arquivo de template `quote-php-template.json` fornecido.

- 4.1. Observe as definições e configurações dos pods, serviços e solicitações de volume persistente definidas no template.

```
[student@workstation quote-php]$ cd ~/D0180/labs/multicontainer-review
```

5. Faça upload do template de aplicativo PHP para que qualquer desenvolvedor com acesso ao seu projeto possa usá-lo.

- 5.1. Use o comando `oc create -f` para fazer upload do arquivo de template no projeto.

```
[student@workstation multicontainer-review]$ oc create -f quote-php-template.json
template.template.openshift.io/quote-php-persistent created
```

6. Processe o template carregado, crie os recursos de aplicativo e confirme o status.

- 6.1. Use o comando `oc process` para processar o arquivo de template. Certifique-se de fornecer o parâmetro `RHT_OCP4_QUAY_USER=${RHT_OCP4_QUAY_USER}` com o namespace `quay.io` no qual as imagens estão localizadas. Use o comando `pipe` para enviar o resultado para o comando `oc create` a fim de criar um aplicativo a partir do template.

```
[student@workstation multicontainer-review]$ oc process quote-php-persistent \
> -p RHT_OCP4_QUAY_USER=${RHT_OCP4_QUAY_USER} \
> | oc create -f -
pod/mysql created
pod/quote-php created
service/quote-php created
```

```
service/mysql created
persistentvolumeclaim/dbinit created
persistentvolumeclaim/dbclaim created
```

- 6.2. Verifique o status da implantação usando o comando `oc get pods` com a opção `-w` para monitorar o status da implantação. Aguarde até que ambos os pods estejam em execução. Pode levar algum tempo para que ambos os pods sejam iniciados.

```
[student@workstation multicontainer-review]$ oc get pods -w
NAME      READY   STATUS            RESTARTS   AGE
mysql     0/1     ContainerCreating   0          21s
quote-php 0/1     ContainerCreating   0          20s
quote-php  1/1    Running           0          35s
mysql     1/1    Running           0          49s
^C
```

Pressione `Ctrl+C` para sair do comando.

7. Exponha o serviço quote-php.

Permita que o aplicativo PHP Quote fique acessível através do roteador do OpenShift e a partir de uma rede externa.

- 7.1. Use o comando `oc expose` para expor o serviço quote-php.

```
[student@workstation multicontainer-review]$ oc expose svc quote-php
route.route.openshift.io/quote-php exposed
```

8. Teste o aplicativo com curl e verifique se ele envia uma mensagem inspiradora.

- 8.1. Use o comando `oc get route` para encontrar o FQDN no qual o aplicativo está disponível. Observe o FQDN do aplicativo.

Execute o comando a seguir na janela de terminal.

```
[student@workstation multicontainer-review]$ oc get route
NAME      HOST/PORT                               PATH  SERVICES   ...
quote-php  quote-php-your_dev_user-deploy.wildcard_domain  quote-php ...
```

- 8.2. Altere para o diretório /home/student.

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

- 8.3. Use o comando `curl` para testar a API REST para o aplicativo PHP Quote.

```
[student@workstation ~]$ curl -w "\n" \
> http://quote-php-${RHT_OCP4_DEV_USER}-deploy.${RHT_OCP4_WILDCARD_DOMAIN}
Always remember that you are absolutely unique. Just like everyone else.
```



nota

O texto exibido na saída acima pode ser diferente, mas o comando `curl` deve ser executado com êxito.

Avaliação

Avalie seu trabalho executando o comando `lab multicontainer-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

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

Encerramento

Na `lab multicontainer-review finish`, execute o comando `workstation` para concluir este laboratório.

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

Isso conclui o laboratório.

Sumário

Neste capítulo, você aprendeu que:

- Redes definidas por software permitem a comunicação entre os contêineres. Os contêineres devem ser anexados à mesma rede definida por software para que se comuniquem.
- Aplicativos conteinerizados não podem contar com endereços IP ou nomes de host fixos para encontrar serviços.
- O Podman usa a Container Network Interface (CNI) para criar uma rede definida por software e anexa todos os contêineres no host a essa rede. O Kubernetes e o OpenShift criam uma rede definida por software entre todos os contêineres em um pod.
- Dentro do mesmo projeto, o Kubernetes injeta um conjunto de variáveis para cada serviço em todos os pods.
- Os modelos do OpenShift automatizam a criação de aplicativos que são compostos por vários recursos. Os parâmetros de template permitem usar os mesmos valores ao criar vários recursos.

capítulo 8

Solução de problemas de aplicativos conteinerizados

Meta

Solucionar os problemas de um aplicativo em contêineres implantado no OpenShift.

Objetivos

- Solucionar os problemas de compilação e implantação de um aplicativo no OpenShift.
- Implementar técnicas para solucionar problemas e depurar aplicativos em contêineres

Seções

- Solução de problemas de compilações e implantações S2I (e exercício orientado)
- Solução de problemas de aplicativos em contêineres (e exercício orientado)

Laboratório

- Solução de problemas de aplicativos conteinerizados

Solução de problemas de builds e implantações S2I

Objetivos

Depois de concluir esta seção, você deverá ser capaz de:

- Solucionar problemas de compilação de um aplicativo e as etapas de implantação no OpenShift.
- Analisar logs do OpenShift para identificar problemas durante o processo de compilação e implantação.

Introdução ao processo S2I

O processo Source-to-Image (S2I) é uma maneira simples de criar imagens automaticamente com base em linguagens de programação do código-fonte do aplicativo no OpenShift. Embora esse processo seja, com frequência, um modo conveniente de implantar aplicativos, podem surgir problemas durante o processo de criação de imagem S2I devido às características da linguagem de programação ou devido ao ambiente de tempo de execução que exige que desenvolvedores e administradores trabalhem juntos.

É importante entender o fluxo de trabalho básico para a maioria das linguagens de programação suportadas pelo OpenShift. O processo de criação de imagem S2I é composto por duas etapas principais:

- Etapa de compilação: responsável por compilar o código-fonte, baixar dependências de biblioteca e empacotar o aplicativo como uma imagem de contêiner. Além disso, a etapa de compilação envia a imagem ao registro do OpenShift para a etapa de implantação. Os recursos `BuildConfig(BC)` do OpenShift conduzem a etapa de compilação.
- Etapa de implantação: responsável por iniciar um pod e disponibilizar o aplicativo para o OpenShift. Essa etapa é executada depois da etapa de compilação, mas somente se a etapa de compilação tiver sido bem-sucedida. Os recursos `Deployment` do OpenShift conduzem a etapa de implantação.

Para o processo S2I, cada aplicativo usa seus próprios objetos `BuildConfig` e `Deployment`, o nome dos quais corresponde ao nome do aplicativo. O processo de implantação é anulado se a compilação falhar.

O processo S2I inicia cada etapa em um pod separado. O processo de compilação cria um pod denominado `<application-name>-build-<number>-<string>`. Em cada tentativa de compilação, toda a etapa de compilação é executada e salva um log. Após uma compilação bem-sucedida, o aplicativo inicia em um pod separado, chamado `<application-name>-<string>`.

É possível usar o console da web do OpenShift para acessar detalhes de cada etapa. Para identificar qualquer problema de compilação, os logs de uma compilação podem ser avaliados e analisados clicando no link `Builds` do painel esquerdo, representado conforme a seguir.

Figura 8.1: Instâncias de compilação de um projeto

Em cada tentativa de compilação, um histórico da compilação, marcado com um número, é fornecido para avaliação. Clicar no nome da compilação leva às páginas de detalhes da compilação:

Figura 8.2: Visão detalhada de uma instância de compilação

Em cada tentativa de compilação, um histórico da compilação, marcado com um número, é fornecido para avaliação. Clicar no nome da compilação leva às páginas de detalhes da compilação.

A guia **Logs** da página de detalhes da compilação mostra a saída gerada pela execução da compilação. Esses logs são úteis para identificar problemas de compilação.

Use o link **Deployment** na seção **Workloads** do painel esquerdo para identificar problemas durante a etapa de implantação.

Depois de selecionar o objeto de implantação apropriado, os detalhes são mostrados na seção **Details**.

A interface de linha de comando oc têm vários subcomandos para gerenciar os logs. Assim como na interface web, ela possui um conjunto de comandos que fornecem informações sobre cada etapa. Por exemplo, para recuperar os logs de uma configuração de compilação, execute o seguinte comando:

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

Se uma compilação falhar, após localizar e corrigir os problemas, execute o seguinte comando para solicitar uma nova compilação:

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

Emitindo esse comando, o OpenShift automaticamente gera um novo pod com o processo de compilação.

Os logs de implantação podem ser verificados com o comando oc:

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

Se a implantação estiver em execução ou tiver falhado, o comando retornará os logs do processo de implantação. Caso contrário, o comando retornará os logs do pod do aplicativo.

Descrição de problemas comuns

Às vezes, o código-fonte exige algumas personalizações que talvez não estejam disponíveis em ambientes conteinerizados, como credenciais de banco de dados, acesso de sistema de arquivos, ou informações de fila de mensagens. Esses valores geralmente assumem a forma de variáveis de ambiente interno. Desenvolvedores usando o processo S2I podem precisar acessar essas informações.

O comando oc logs fornece informações importantes sobre a compilação, a implantação e os processos de execução de um aplicativo durante a execução de um pod. Os logs podem indicar valores ausentes ou opções que devem ser habilitadas, parâmetros ou sinalizadores incorretos ou incompatibilidades de ambiente.



nota

Logs de aplicativos devem ser claramente rotulados para identificar problemas rapidamente sem a necessidade de aprender as características internas do contêiner.

Solução de problemas de permissão

O OpenShift executa contêineres S2I usando o Red Hat Enterprise Linux como a imagem de base, e qualquer diferença de tempo de execução pode levar a uma falha no processo S2I. Algumas vezes, o desenvolvedor se depara com problemas de permissão, como uma negação de acesso devido a permissões erradas ou permissões de ambiente incorretas definidas pelos administradores. Imagens S2I reforçam o uso de um usuário diferente do usuário root para acessar sistemas de arquivos e recursos externos. Além disso, o Red Hat Enterprise Linux 8 impõe as políticas do SELinux que restringem o acesso a alguns recursos de sistema de arquivos, portas de rede ou processos.

Alguns contêineres podem precisar de uma ID de usuário específica. Já o S2I é projetado para executar contêineres usando um usuário aleatório conforme a política de segurança padrão do OpenShift.

O Dockerfile a seguir cria um contêiner Nexus. Observe que a instrução USER indica que o usuário nexus seja usado:

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

USER nexus
WORKDIR ${NEXUS_HOME}

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

Tentar usar a imagem gerada por esse Dockerfile sem endereçar permissões de volume leva a erros quando o contêiner é iniciado:

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

Para resolver esse problema, amenize a segurança do projeto do OpenShift com o comando `oc adm policy`:

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

O comando `oc adm policy` permite que o OpenShift execute processos de contêiner com usuários não root. Mas os sistemas de arquivos usados no contêiner também devem estar disponíveis para o usuário em execução. Isso é especialmente importante quando o contêiner contém montagens de volume.

Para evitar problemas de permissão do sistema de arquivos, as pastas locais usadas para montagens de volume do contêiner devem satisfazer o seguinte:

- O usuário que está executando os processos de contêiner deve ser o proprietário da pasta ou ter os direitos necessários. Use o comando `chown` para atualizar a propriedade da pasta.
- A pasta local deve satisfazer os requisitos do SELinux a serem usados como um volume de contêiner. Atribua o grupo `container_file_t` à pasta usando o comando `semanage fcontext -a -t container_file_t <folder>`; depois, atualize as permissões com o comando `restorecon -R <folder>`.

Solução de problemas de parâmetros inválidos

Aplicativos de vários contêineres podem compartilhar parâmetros, como credenciais de login. Certifique-se de que os mesmos valores dos parâmetros cheguem a todos os contêineres no aplicativo. Por exemplo, para um aplicativo Python executado em um contêiner conectado a outro contêiner executando um banco de dados, certifique-se de que os dois contêineres usam os mesmos nome de usuário e senha para o banco de dados. Geralmente, os logs do pod do aplicativo fornecem uma ideia clara desses problemas e como resolvê-los.

Uma prática recomendada para centralizar os parâmetros compartilhados é armazená-los em `ConfigMaps`. Esse `ConfigMaps` pode ser injetado através do `Deployment` em contêineres

capítulo 8 | Solução de problemas de aplicativos conteinerizados

como variáveis de ambiente. Injetando o mesmo ConfigMap em contêineres diferentes garante que não apenas as mesmas variáveis de ambiente estejam disponíveis, mas também os mesmos valores. Veja a seguinte definição de recurso de pods:

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

- ❶ Uma variável de ambiente ENV_1 é injetada no contêiner. Seu valor é o valor para a entrada configMap_key_1 no configMap configMap_name1.
- ❷ Todas as entradas em configMap_name_2 são injetadas no contêiner como variáveis de ambiente com o mesmo nome e os mesmos valores.

Solução de problemas de erros de montagem de volume

Ao reimplantar um aplicativo que usa um volume persistente em um sistema de arquivos local, um pod pode não ser capaz de alocar uma solicitação de volume persistente, embora o volume persistente indique que a solicitação está liberada.

Para resolver o problema, exclua a solicitação de volume persistente e, depois, o volume persistente. Então, recrie o volume persistente:

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

Solução de problemas de imagens obsoletas

O OpenShift faz pull de imagens da fonte indicada em um fluxo de imagem, a menos que ele encontre uma imagem localmente em cache no nó em que o pod está agendado para ser executado. Se você enviar uma nova imagem ao registro com os mesmos nome e tag, deverá remover a imagem de cada nó em que o pod está agendado com o comando `podman rmi`.

Execute o comando `oc adm prune` para ver uma maneira automatizada de remover imagens obsoletas e outros recursos.



Referências

Mais informações sobre a solução de problemas de imagens estão disponíveis na seção *Images* da documentação do OpenShift Container Platform, disponível em:

Criação de imagens

https://docs.openshift.com/container-platform/4.10/openshift_images/create-images.html

A documentação sobre como consumir o ConfigMap para criar variáveis de ambiente de contêiner pode ser encontrada em *Consuming in Environment Variables* no

Configuração de um pod para usar o ConfigMaps

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

► Exercício Guiado

Solução de problemas de uma build do OpenShift

Neste exercício, você solucionará problemas de uma compilação do OpenShift e de um processo de implantação.

Resultados

Você deverá ser capaz de identificar e resolver os problemas que surgiram durante o processo de build e de implantação de um aplicativo Node.js.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Recupere os arquivos de laboratório e verifique, executando o comando a seguir, se o Docker e o cluster do OpenShift estão em execução.

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

Instruções

- 1. Carregue a configuração do seu ambiente de sala de aula. Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- 2. Digite seu clone local do repositório git D0180-apps e faça check-out do branch master do repositório do curso para garantir que você inicie este exercício em um estado que tenha boas condições:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 3. Crie um novo branch para salvar as alterações feitas durante este exercício:

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

- 4. Faça login no OpenShift usando o usuário configurado, a senha e a URL da API mestre.

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

Crie um novo projeto chamado *youruser-nodejs*.

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

- 5. Compile um novo aplicativo Node.js usando a imagem Hello World localizada em <https://github.com/yourgituser/D0180-apps/> no diretório nodejs-helloworld.

- 5.1. Execute o comando `oc new-app` para criar o aplicativo Node.js. O comando é fornecido no arquivo `~/D0180/labs/troubleshoot-s2i/command.txt`.

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

Node.js 16
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "nodejs-hello" created
buildconfig.build.openshift.io "nodejs-hello" created
deployment.apps "nodejs-hello" created
service "nodejs-hello" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/nodejs-hello' to track its
progress.
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose svc/nodejs-hello'
Run 'oc status' to view your app.
```

O `-i` indica a imagem da compilação a ser usada, `nodejs:12` neste caso.

A opção `--context-dir` define qual pasta dentro do projeto contém o código-fonte do aplicativo a ser compilado.

A opção `--build-env` define uma variável de ambiente no pod de construtor. Nesse caso, ele fornece a variável de ambiente `npm_config_registry` para o pod de construtor, para que ele possa acessar o registro do NPM.

**Importante**

No comando anterior, não deve haver espaços entre `registry=` e a URL do servidor Nexus.

- 5.2. Aguarde até que o aplicativo conclua a criação monitorando o progresso com o comando `oc get pods -w`. O pod faz a transição de um status `running` para `Error`:

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

O processo de compilação falha e, portanto, nenhum aplicativo está em execução. Normalmente, falhas de compilação são consequência de erros de sintaxe no código-fonte ou dependências ausentes. A próxima etapa investiga as causas específicas para essa falha.

- 5.3. Avalie os erros ocorridos durante o processo de compilação.

A compilação é acionada pela configuração de compilação (`bc`) criada pelo OpenShift quando o processo S2I é iniciado. Por padrão, o processo S2I do OpenShift cria uma configuração de compilação denominada conforme o nome definido: `nodejs-hello`, que é responsável por acionar o processo de compilação.

Execute o comando `oc` com o subcomando `logs` em uma janela de terminal para confirmar a saída do processo de build:

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

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

...output omitted...
```

O log mostra um erro durante o processo de criação. Esta saída indica que não há versão compatível para a dependência `express`. Mas o motivo é que o formato usado pela dependência `express` não é válido.

► 6. Atualize o processo de compilação do projeto.

O desenvolvedor usa uma versão não padrão da estrutura Express que está disponível localmente na estação de trabalho de cada desenvolvedor. Devido aos padrões da empresa, a versão deve ser baixada do registro oficial Node.js e, a partir da entrada do desenvolvedor, é compatível com a versão 4.14.x.

6.1. Corrija o arquivo `package.json`.

Use o editor de sua preferência para abrir o arquivo `~/D0180-apps/nodejs-helloworld/package.json`. Revise as versões da dependência fornecidas pelos desenvolvedores. Ela usa uma versão incorreta da dependência Express, que é incompatível com a versão suportada fornecida pela empresa (~4.14.2). Atualize a versão da dependência da seguinte maneira.

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



nota

Observe o x na versão. Indica que a maior versão deve ser usada, mas a versão deve começar com 4.14..

6.2. Faça commit das alterações feitas e envie-as ao projeto.

Na janela de terminal, execute o seguinte comando para fazer commit e enviar as alterações:

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

► 7. Reinicie o processo S2I.

7.1. Para reiniciar a etapa de compilação, execute o seguinte comando:

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

A etapa de compilação é reiniciada e um novo pod de compilação é criado. Verifique o log executando o comando `oc logs`.

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

A compilação teve êxito. Entretanto, isso não indica que o aplicativo foi iniciado.

- 7.2. Avalie o status do processo atual de compilação. Execute o comando `oc get pods` para verificar o status do aplicativo Node.js.

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

De acordo com a saída a seguir, a segunda build foi concluída, mas o aplicativo está em um estado de erro.

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

O nome do pod de aplicativo (`nodejs-hello-7b99966464-fw4r8`) é gerado aleatoriamente e pode ser diferente do seu.

- 7.3. Revise os logs gerados pelo pod do aplicativo.

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



nota

O comando `oc logs nodejs-hello-7b99966464-fw4r8` despeja os logs do pod de implantação. No caso de uma implantação bem-sucedida, esse comando despeja os logs do pod do aplicativo, conforme mostrado anteriormente.

Se os logs de implantação não exibirem o erro, verifique os logs do pod do aplicativo executando o comando `oc logs POD` substituindo `POD` pelo nome do pod que tem um status de `CrashLoopBackOff`.

Ocorre uma falha na inicialização do aplicativo porque a declaração do script de início está ausente.

- 8. Corrija o problema atualizando o código do aplicativo.

- 8.1. Atualize o arquivo `package.json` para definir um comando de inicialização.

A saída anterior indica que o arquivo `~/D0180-apps/nodejs-helloworld/package.json` não contém o atributo `start` no campo `scripts`. O atributo `start` define um comando a ser executado quando o aplicativo é iniciado. Ele invoca o binário `node`, que executa o aplicativo `app.js`.

Para corrigir o problema, adicione o atributo a seguir ao arquivo `package.json`. Não se esqueça da vírgula após o colchete.

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

8.2. Confirme as alterações feitas e envie-as ao projeto:

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

Continua com a etapa de implantação do processo S2I.

8.3. Reinicie a etapa de compilação.

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

8.4. Avalie o status do processo atual de compilação. Execute o comando para recuperar o status do aplicativo Node.js. Aguarde até que a compilação mais recente seja concluída.

NAME	READY	STATUS	RESTARTS	AGE
nodejs-hello-1-build	0/1	Error	0	26m
nodejs-hello-2-build	0/1	Completed	0	19m
nodejs-hello-3-build	1/1	Running	0	9s
nodejs-hello-7b99966464-fw4r8	0/1	CrashLoopBackOff	8	19m
nodejs-hello-6dcd88b7b7-dvtcz	0/1	Pending	0	0s
nodejs-hello-6dcd88b7b7-dvtcz	0/1	Pending	0	0s
nodejs-hello-6dcd88b7b7-dvtcz	0/1	ContainerCreating	0	0s
nodejs-hello-3-build	0/1	Completed	0	37s
nodejs-hello-6dcd88b7b7-dvtcz	0/1	ContainerCreating	0	2s
nodejs-hello-6dcd88b7b7-dvtcz	1/1	Running	0	3s
nodejs-hello-7b99966464-fw4r8	0/1	Terminating	8	19m
nodejs-hello-7b99966464-fw4r8	0/1	Terminating	8	19m
nodejs-hello-7b99966464-fw4r8	0/1	Terminating	8	19m

De acordo com a saída, a compilação foi bem-sucedida e o aplicativo foi inicializado sem erros. A saída também fornece informações sobre como o pod de

capítulo 8 | Solução de problemas de aplicativos conteinerizados

implantação (`nodejs-hello-3-build`) foi criado, e de que ele foi concluído e encerrado com êxito. Como o novo pod do aplicativo está disponível (`nodejs-hello-7b99966464-dvtcz`), o antigo (`nodejs-hello-7b99966464-fw4r8`) é lançado.

- 8.5. Revise os logs gerados pelo pod do aplicativo `nodejs-hello`.

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

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

Example app listening on port 8080!
```

O aplicativo agora está em execução na porta 8080.

► 9. Teste o aplicativo.

- 9.1. Execute o comando `oc` com o subcomando `expose` para expor o aplicativo.

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

- 9.2. Recupere o endereço associado ao aplicativo.

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

- 9.3. Acesse o aplicativo na VM `workstation` usando o comando `curl`:

```
[student@workstation D0180-apps]$ curl -w "\n" \
> http://nodejs-hello-${RHT_OCP4_DEV_USER}-nodejs.${RHT_OCP4_WILDCARD_DOMAIN}
Hello world!
```

A saída demonstra que o aplicativo está ativo e em execução.

Encerramento

Na workstation, execute o script `lab troubleshoot-s2i finish` para concluir esse exercício.

```
[student@workstation D0180-apps]$ lab troubleshoot-s2i finish
```

Isso conclui o exercício.

Solução de problemas de aplicativos conteinerizados

Objetivos

Depois de concluir esta seção, você deverá ser capaz de:

- Implementar técnicas para solucionar problemas e depurar aplicativos em contêineres
- Usar o recurso de encaminhamento de porta da ferramenta do cliente do OpenShift.
- Visualizar logs de registro de contêineres.
- Visualizar eventos de cluster do OpenShift.

Encaminhamento de portas para solução de problemas

Às vezes, os desenvolvedores e os administradores de sistema precisam de acesso especial à rede para um contêiner que não seria necessário para usuários do aplicativo. Por exemplo, talvez os desenvolvedores precisem usar o console de administração para um banco de dados ou serviço de mensagens, ou talvez os administradores do sistema façam uso do acesso SSH a um contêiner para reiniciar um serviço encerrado. Esse acesso à rede, na forma de portas de rede, geralmente não é exposto pelas configurações de contêiner padrão e tende a exigir clientes especializados usados por desenvolvedores e administradores de sistema.

O Podman fornece recursos de encaminhamento de porta usando a opção `-p` com o subcomando `run`. Nesse caso, não há distinções entre o acesso à rede para acesso regular de aplicativo e para solução de problemas. Para relembrar, abaixo temos um exemplo de configuração de encaminhamento de porta por meio do mapeamento da porta do host para um servidor de banco de dados em execução em um contêiner:

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

O comando anterior mapeia a porta 30306 do host para a porta 3306 no contêiner `db`. Esse contêiner é criado a partir da imagem `mysql`, que inicia um servidor MySQL que escuta a porta 3306.

O OpenShift oferece o comando `oc port-forward` para encaminhar uma porta local para uma porta de pod. Isso é diferente de ter acesso a um pod através de um recurso de serviço:

- O mapeamento de encaminhamento de porta existe somente na estação de trabalho na qual o cliente `oc` é executado, enquanto um serviço mapeia uma porta para todos os usuários da rede.
- Um serviço balanceia a carga de conexões para pods potencialmente múltiplos, enquanto um mapeamento de encaminhamento de porta encaminha conexões para um único pod.

Aqui está um exemplo do comando `oc port-forward`:

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

O comando anterior encaminha a porta 30306 da máquina do desenvolvedor para a porta 3306 no pod db, onde um servidor MySQL (dentro de um contêiner) aceita conexões de rede.

**nota**

Ao executar esse comando, certifique-se de deixar a janela de terminal em execução. Fechar a janela ou cancelar o processo interrompe o mapeamento de porta.

Embora o método `podman run -p` de mapeamento de porta (encaminhamento de porta) só possa ser configurado quando o contêiner for iniciado, o mapeamento com o comando `oc port-forward` pode ser criado e destruído a qualquer momento após a criação de um pod.

**nota**

A criação de um serviço do tipo `NodePort` para um pod de banco de dados seria semelhante a executar `podman run -p`. Entretanto, a Red Hat desencoraja o uso da abordagem `NodePort` para evitar expor o serviço a conexões diretas. Mapear com encaminhamento de porta no OpenShift é considerado uma alternativa mais segura.

Ativação da depuração remota com o encaminhamento de porta

Outro uso para o recurso de encaminhamento de porta é a ativação da depuração remota. Muitos ambientes integrados de desenvolvimento (IDEs) oferecem a capacidade de depurar remotamente um aplicativo.

For example, Red Hat CodeReady Studio allows users to utilize the Java Debug Wire Protocol (JDWP) to communicate between its debugger and the Java Virtual Machine. Quando ele estiver habilitado, os desenvolvedores podem percorrer todas as linhas do código à medida que ele estiver sendo executado em tempo real.

Para o JDWP funcionar, a máquina virtual Java (JVM) onde o aplicativo é executado deve ser iniciada com opções que habilitam a depuração remota. Por exemplo, os usuários do WildFly e do JBoss EAP precisam configurar essas opções na inicialização do servidor de aplicativos. A linha abaixo no arquivo `standalone.conf` possibilita a depuração remota abrindo a porta JDWP TCP 8787 para uma instância do WildFly ou EAP em execução no modo independente:

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

Quando o servidor é iniciado com o depurador ouvindo na porta 8787, um mapeamento de porta precisa ser criado para encaminhar conexões de uma porta TCP local não utilizada para a porta 8787 no pod do EAP. Se a estação de trabalho do desenvolvedor não tiver nenhuma JVM local em execução com depuração remota ativada, a porta local também poderá ser 8787.

O comando abaixo presume que há um pod do WildFly denominado `jappserver` em execução em um contêiner a partir de uma imagem previamente configurada para permitir a depuração remota:

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

capítulo 8 | Solução de problemas de aplicativos conteinerizados

Assim que o depurador estiver ativado e o mapeamento de encaminhamento de porta for criado, os usuários poderão definir pontos de interrupção em seu IDE de escolha e executar o depurador apontando para o nome de host do aplicativo e a porta de depuração (nesta instância, 8787).

Acesso a logs de contêiner

O Podman e o OpenShift oferecem a capacidade de visualizar logs em contêineres e pods em execução para facilitar a solução de problemas. Porém, nenhum deles detecta os logs específicos do aplicativo. Ambos esperam que o aplicativo seja configurado para enviar toda a saída de registros de log a uma saída padrão.

Um contêiner é simplesmente uma árvore de processos na perspectiva do SO do host. Quando o Podman inicia um contêiner diretamente ou no cluster do RHOC, ele redireciona a saída padrão do contêiner e o erro padrão, salvando-os no disco como parte do armazenamento efêmero do contêiner. Dessa forma, os logs do contêiner podem ser visualizados através dos comandos `podman` e `oc`, mesmo depois do contêiner ser interrompido, mas não removido.

Para recuperar a saída de um contêiner em execução, use o comando `podman` a seguir.

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

No OpenShift, o comando a seguir retorna a saída para um contêiner dentro de um pod:

```
$ oc logs <podName> [-c <containerName>]
```



nota

O nome do contêiner é opcional se houver apenas um contêiner, pois `oc` usa como padrão o único contêiner em execução e retorna a saída.

Eventos do OpenShift

Alguns desenvolvedores consideram os logs do Podman e do OpenShift como sendo de baixo nível, o que dificulta a solução de problemas. Felizmente, o OpenShift oferece um registro de log de nível superior e um recurso de auditoria denominado eventos.

Os eventos do OpenShift sinalizam ações significativas como a inicialização de um contêiner ou a destruição de um pod.

Para ler eventos do OpenShift, use o subcomando `get` com o tipo de recurso `events` para o comando `oc` conforme a seguir.

```
$ oc get events
```

Eventos listados pelo comando `oc` dessa forma não são filtrados e abrangem todo o cluster do RHOC. Usar um pipe para filtros UNIX padrão, como `grep`, pode ajudar, mas o OpenShift oferece uma alternativa para consultar eventos do cluster. A abordagem é fornecida pelo subcomando `describe`.

Por exemplo, para somente recuperar os eventos que se relacionam a um pod `mysql`, consulte o campo `Events` da saída do comando `oc describe pod mysql`.

```
$ oc describe pod mysql
...output omitted...
Events:
FirstSeen   LastSeen   Count  From           Reason          Message
Wed, 10 ... Wed, 10 ... 1    {scheduler } scheduled  Successfully as...
...output omitted...
```

Acesso a contêineres em execução

Os comandos `podman logs` e `oc logs` podem ser úteis para ver a saída enviada pelo contêiner. Entretanto, a saída não necessariamente exibe todas as informações disponíveis se o aplicativo estiver configurado para enviar logs a um arquivo. Outros cenários de solução de problemas podem exigir a inspeção do ambiente do contêiner, conforme visto por processos dentro do contêiner, como verificar a conectividade externa.

Como solução, o Podman e o OpenShift oferecem o subcomando `exec`, permitindo a criação de novos processos dentro de um contêiner em execução, com a saída e a entrada padrão desses processos redirecionadas ao terminal do usuário. A seguinte tela exibe o uso do comando `podman exec`:

```
$ podman exec [options] container command [arguments]
```

A sintaxe geral para o comando `oc exec` é:

```
$ oc exec [options] pod [-c container] -- command [arguments]
```

Para executar um único comando interativo ou iniciar um shell, adicione as opções `-it`. O exemplo abaixo inicia um shell Bash para o pod `myhttpd`:

```
$ oc exec -it myhttpd /bin/bash
```

Você pode usar esse comando para acessar logs do aplicativo salvos no disco (como parte do armazenamento efêmero do contêiner). Por exemplo, para exibir o log de erro do Apache de um contêiner, execute o seguinte comando:

```
$ podman exec apache-container cat /var/log/httpd/error_log
```

Substituição de binários de contêiner

Muitas imagens de contêiner não contêm todos os comandos de solução de problemas que os usuários esperam encontrar em instalações regulares de SO, como `telnet`, `netcat`, `ip` ou `traceroute`. Retirar utilitários básicos ou binários da imagem permite que ela permaneça leve, executando muitos contêineres por host.

Uma maneira de acessar temporariamente alguns desses comandos ausentes é montar as pastas de binários de host (como `/bin`, `/sbin` e `/lib`) como volumes dentro do contêiner. Isso é possível porque a opção `-v` do comando `podman run` não exige instruções `VOLUME` correspondentes para estar presente no `Containerfile` da imagem de contêiner.

**nota**

Para acessar esses comandos do OpenShift, você precisa alterar a definição do recurso do pod para definir os objetos `volumeMounts` e `volumeClaims`. Também é necessário criar um volume persistente `hostPath`.

O comando a seguir inicia um contêiner e substitui a pasta `/bin` da imagem pela pasta do host. Ele também inicia um shell interativo dentro do contêiner:

```
$ podman run -it -v /bin:/bin image /bin/bash
```

**nota**

O diretório de binários a ser substituído depende da imagem de SO de base. Por exemplo, alguns comandos precisam de bibliotecas compartilhadas do diretório `/lib`. Algumas distribuições do Linux têm diferentes conteúdos em `/bin`, `/usr/bin`, `/lib` ou `/usr/lib`, o que exigiria o uso da opção `-v` para cada diretório.

Como uma alternativa, você pode incluir esses utilitários na imagem base. Para fazer isso, adicione instruções em uma definição de compilação do `Containerfile`. Por exemplo examine o seguinte trecho de uma definição do `Containerfile`, que é filho da imagem `rhel7.5`. A instrução `RUN` instala as ferramentas que são comumente usadas para solucionar problemas de rede:

```
FROM ubi7/ubi:7.7

RUN yum install -y \
    less \
    dig \
    ping \
    iputils && \
    yum clean all
```

Quando a imagem for compilada e o contêiner for criado, ela será idêntica à imagem de contêiner `rhel7.5`, mas as ferramentas adicionais disponíveis.

Inclusão e remoção de arquivos de contêineres

Ao solucionar problemas de um aplicativo ou gerenciá-lo, você pode precisar recuperar ou transferir arquivos de ou para contêineres em execução, como arquivos de configuração ou arquivos de log. Há várias maneiras de mover arquivos para e de um contêiner, como descrito na lista a seguir.

Montagens de volume

Outra opção para copiar arquivos de um host para um contêiner é o uso de montagens de volume. Você pode montar um diretório local para copiar dados para um contêiner. Por exemplo, o seguinte comando define o diretório de host `/conf` como o volume a ser usado para o diretório de configuração do Apache. Isso oferece uma maneira conveniente de gerenciar o servidor Apache sem ter que recompilar a imagem de contêiner.

```
$ podman run -v /conf:/etc/httpd/conf -d do180/apache
```

podman cp

O subcomando `cp` permite aos usuários copiar os arquivos de um contêiner em execução ou para ele. Para copiar um arquivo para um contêiner chamado `todoapi`, execute o comando a seguir.

```
$ podman cp standalone.conf todoapi:/opt/jboss/standalone/conf/standalone.conf
```

Para copiar um arquivo de um contêiner para o host, altere a ordem do comando anterior.

```
$ podman cp todoapi:/opt/jboss/standalone/conf/standalone.conf .
```

O comando `podman cp` tem a vantagem de funcionar com contêineres que já foram iniciados, enquanto a alternativa a seguir (montagens de volume) precisa de alterações no comando usado para iniciar um contêiner.

podman exec

Para os contêineres que já estão em execução, o comando `podman exec` pode ser conectado para passar arquivos a partir de um contêiner em execução ou para um contêiner em execução por meio da adição de comandos que são executados no contêiner. O exemplo a seguir mostra como passar e executar um arquivo SQL dentro de um contêiner do MySQL:

```
$ podman exec -i <container> mysql -uroot -proot < /path/on/host/db.sql
```

Usando o mesmo conceito, é possível recuperar dados de um contêiner em execução e colocá-los na máquina do host. Um exemplo útil desse uso do utilitário `mysqldump`, que cria um backup do banco de dados MySQL do contêiner e colocá-lo no host.

```
$ podman exec -it <containerName> sh \
> -c 'exec mysqldump -h"$MYSQL_PORT_3306_TCP_ADDR" \
> -P"$MYSQL_PORT_3306_TCP_PORT" \
> -uroot -p"$MYSQL_ENV_MYSQL_ROOT_PASSWORD" items' > db_dump.sql
```

O comando anterior usa as variáveis do ambiente de contêiner para se conectar ao servidor MySQL, executar o comando `mysqldump` e redirecionar a saída para um arquivo na máquina de host. Ele presume que a imagem de contêiner fornece o utilitário `mysqldump`, então, não há necessidade de instalar as ferramentas de administração do MySQL no host.

O comando `oc rsync` oferece recursos semelhantes ao `podman cp` para contêineres em execução nos pods do OpenShift.



Referências

Mais informações sobre o encaminhamento de porta estão disponíveis na seção *Port Forwarding* da documentação do OpenShift Container Platform em

Arquitetura

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/architecture/index/

Mais informações sobre comandos da CLI para encaminhamento de porta estão disponíveis no capítulo *Port Forwarding* da documentação do OpenShift Container Platform em

Desenvolvimento de aplicativos

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.10/html-single/building_applications/index

► Exercício Guiado

Configuração de logs de contêiner do Apache para depuração

Neste exercício, você configurará um contêiner httpd o Apache para enviar os logs ao `stdout`; depois, você revisará os logs e eventos do Podman.

Resultados

Você deverá ser capaz de configurar um contêiner httpd do Apache para enviar logs de depuração ao `stdout` e visualizá-los com o comando `podman logs`.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Recupere os arquivos de laboratório e verifique, executando o comando a seguir, se o Docker e o cluster do OpenShift estão em execução.

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

Instruções

- 1. Configure o servidor web do Apache para enviar mensagens de log à saída padrão e atualize o nível de log padrão.
 - 1.1. O nível de log padrão para a imagem httpd do Apache é `warn`. Altere o nível de log padrão do contêiner para `debug` e redirecione as mensagens de log para `stdout`, substituindo o arquivo de configuração padrão `httpd.conf`. Para fazer isso, crie uma imagem personalizada da VM `workstation`.
Revise brevemente o arquivo personalizado `httpd.conf` localizado em `/home/student/D0180/labs/troubleshoot-container/conf/httpd.conf`.
 - Observe a diretiva `ErrorLog` no arquivo:

```
ErrorLog "/dev/stdout"
```

A diretiva envia mensagens de log de erro httpd à saída padrão do contêiner.

- Observe a diretiva `LogLevel` no arquivo.

```
LogLevel debug
```

A diretiva altera o nível do log padrão para `debug`.

- Observe a diretiva `CustomLog` no arquivo.

```
CustomLog "/dev/stdout" common
```

A diretiva redireciona as mensagens de log de acesso httpd para a saída padrão do contêiner.

- 2. Compile um contêiner personalizado para salvar um arquivo de configuração atualizado no contêiner.

- 2.1. Na janela de terminal, execute os comandos a seguir para compilar uma nova imagem.

```
[student@workstation ~]$ cd ~/DO180/labs/troubleshoot-container
[student@workstation troubleshoot-container]$ podman build \
> -t troubleshoot-container .
STEP 1: FROM quay.io/redhattraining/httpd-parent
...output omitted...
STEP 5: COMMIT troubleshoot-container
e23d...c1de
[student@workstation troubleshoot-container]$ cd ~
```

- 2.2. Verifique se a imagem foi criada.

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

A nova imagem deve estar disponível no armazenamento local.

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
localhost/troubleshoot-container	latest	e23df...	9 seconds ago	137MB
quay.io/redhattraining/httpd-parent	latest	0eba3...	4 weeks ago	137MB

- 3. Crie um novo contêiner httpd a partir da imagem personalizada.

```
[student@workstation ~]$ podman run \
> --name troubleshoot-container -d \
> -p 10080:80 troubleshoot-container
4c8bb12815cc02f4eef0254632b7179bd5ce230d83373b49761b1ac41fc067a9
```

- 4. Revise as mensagens e os eventos de log do contêiner.

- 4.1. Visualize as mensagens de log de depuração do contêiner usando o comando `podman logs`:

```
[student@workstation ~]$ podman logs -f troubleshoot-container
... [mpm_event:notice] [pid 1:tid...] AH00489: Apache/2.4.37 ...
... [mpm_event:info] [pid 1:tid...] AH00490: Server built: Apr 5 2019 07:31:21
... [core:notice] [pid 1:tid...] AH00094: Command line: 'httpd -D FOREGROUND'
... [core:debug] [pid 1:tid ...]: AH02639: Using SO_REUSEPORT: yes (1)
... [mpm_event:debug] [pid 6:tid ...]: AH02471: start_threads: Using epoll
... [mpm_event:debug] [pid 7:tid ...]: AH02471: start_threads: Using epoll
... [mpm_event:debug] [pid 8:tid ...]: AH02471: start_threads: Using epoll
```

Observe os logs de depuração, disponíveis na saída padrão.

- 4.2. Abra um novo terminal e acesse a página inicial do servidor web usando o comando curl:

```
[student@workstation ~]$ curl http://127.0.0.1:10080
Hello from the httpd-parent container!
```

- 4.3. Revise as novas entradas no log. Olhe no terminal executando o comando podman logs para ver as novas entradas.

```
...output omitted...
10.0.2.2 - - [31/Mar/2021:14:06:03 +0000] "GET / HTTP/1.1" 200 39
```

- 4.4. Interrompa o comando do Podman com Ctrl+C.

Encerramento

Na workstation, execute o script a seguir para concluir este laboratório.

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

Isso conclui o exercício orientado.

► Laboratório Aberto

Solução de problemas de aplicativos conteinerizados

Resultados

Você deverá ser capaz de identificar e resolver os problemas que surgiram durante o processo de build e de implantação de um aplicativo Node.js.

Antes De Começar

Certifique-se de ter concluído a Exercício Guiado: Configuração do ambiente de sala de aula do Capítulo 1 antes de executar qualquer comando desta prática.

Abra um terminal na workstation como usuário student e execute o seguinte comando:

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

Instruções

- Carregue a configuração do seu ambiente de sala de aula. Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:
 - Digite seu clone local do repositório git D0180-apps e faça check-out do branch master do repositório do curso para garantir que você inicie este exercício em um estado que tenha boas condições:
 - Crie uma nova ramificação chamada troubleshoot-review e envie-a para salvar as alterações feitas durante este exercício:
 - Faça login no OpenShift usando o usuário configurado, a senha e a URL da API mestre.
 - Crie um novo projeto chamado youruser-nodejs-app:
 - No projeto youruser-nodejs-app, crie um novo aplicativo chamado nodejs-dev. Use o fluxo de imagem nodejs:12 e o código-fonte no diretório nodejs-app, localizado no branch do repositório Git troubleshoot-review que você criou em uma etapa anterior. Defina o valor da variável de ambiente npm_config_registry com http:// \${RHT_OCP4_NEXUS_SERVER}/repository/nodejs.
- Espere que o processo de build do aplicativo falhe. Monitore o processo de criação e identifique a falha de compilação.
- Atualize a versão da dependência express no arquivo package.json com um valor 4.x. Faça commit e envie as alterações por push para o repositório Git.
 - Compile o aplicativo novamente. Verifique se o aplicativo é compilado sem erros.
 - Verifique se o aplicativo não está em execução devido a um erro de tempo de execução. Revise os logs e identifique o problema.
 - O log indica que o arquivo server.js tenta carregar um módulo chamado http-error, que não está disponível. Altere o módulo necessário na primeira linha de server.js para html-errors, que é um módulo válido no arquivo packages. Faça commit e envie por push

as alterações do aplicativo para o repositório Git. Compile o aplicativo novamente. Depois que o aplicativo for compilado, verifique se ele está sendo executado.

11. Crie uma rota para o aplicativo e teste o acesso a ele. Espere uma mensagem de erro. Revise os logs para identificar o erro.
12. Substitua `process.environment` por `process.env` no arquivo `server.js` para corrigir o erro. Faça commit e envie as alterações do aplicativo por push para o repositório Git. Compile o aplicativo novamente. Quando o novo aplicativo for implantado, verifique se o aplicativo não gera erros ao acessar o URL do aplicativo.

Avaliação

Avalie seu trabalho executando o comando `lab troubleshoot-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

Encerramento

Na `workstation`, execute o comando `lab troubleshoot-review finish` para concluir este laboratório.

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

Isso conclui o laboratório.

► Solução

Solução de problemas de aplicativos conteinerizados

Resultados

Você deverá ser capaz de identificar e resolver os problemas que surgiram durante o processo de build e de implantação de um aplicativo Node.js.

Antes De Começar

Certifique-se de ter concluído a Exercício Guiado: Configuração do ambiente de sala de aula do Capítulo 1 antes de executar qualquer comando desta prática.

Abra um terminal na workstation como usuário student e execute o seguinte comando:

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

Instruções

- Carregue a configuração do seu ambiente de sala de aula. Execute o seguinte comando para carregar as variáveis de ambiente criadas no primeiro exercício orientado:

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

- Digite seu clone local do repositório git D0180-apps e faça check-out do branch master do repositório do curso para garantir que você inicie este exercício em um estado que tenha boas condições:

```
[student@workstation ~]$ cd ~/D0180-apps  
[student@workstation D0180-apps]$ git checkout master  
...output omitted...
```

- Crie uma nova ramificação chamada troubleshoot-review e envie-a para salvar as alterações feitas durante este exercício:

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

- Faça login no OpenShift usando o usuário configurado, a senha e a URL da API mestre.

```
[student@workstation D0180-apps]$ oc login -u "${RHT_OCP4_DEV_USER}" \
> -p "${RHT_OCP4_DEV_PASSWORD}" "${RHT_OCP4_MASTER_API}"
Login successful.
...output omitted...
```

5. Crie um novo projeto chamado *youruser-nodejs-app*:

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

6. No projeto *youruser-nodejs-app*, crie um novo aplicativo chamado *nodejs-dev*. Use o fluxo de imagem *nodejs:12* e o código-fonte no diretório *nodejs-app*, localizado no branch do repositório Git *troubleshoot-review* que você criou em uma etapa anterior. Defina o valor da variável de ambiente *npm_config_registry* com *http://\${RHT_OCP4_NEXUS_SERVER}/repository/nodejs*.

Espere que o processo de build do aplicativo falhe. Monitore o processo de criação e identifique a falha de compilação.

- 6.1. Execute o comando *oc new-app* para criar o aplicativo Node.js.

```
[student@workstation ~]$ oc new-app --name nodejs-dev \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#troubleshoot-review \
> -i nodejs:16-ubi8 --context-dir=nodejs-app --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/nodejs
--> Found image a2b5ec2 ...output omitted...

Node.js 16
...output omitted...
--> Creating resources ...
imagestream.image.openshift.io "nodejs-dev" created
buildconfig.build.openshift.io "nodejs-dev" created
deployment.apps "nodejs-dev" created
service "nodejs-dev" created
--> Success
Build scheduled, use 'oc logs -f buildconfig/nodejs-dev' to track its
progress.
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
'oc expose service/nodejs-dev'
Run '+oc status' to view your app.
```

- 6.2. Monitore o progresso de build com o comando *oc logs -f bc/nodejs-dev*:

```
[student@workstation ~]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
STEP 8: RUN /usr/libexec/s2i/assemble
--> Installing application source ...
--> Installing all dependencies
npm ERR! code ETARGET
```

```
npm ERR! notarget No matching version found for express@4.20.
npm ERR! notarget In most cases you or one of your dependencies are requesting
npm ERR! notarget a package version that doesn't exist.
npm ERR! notarget
npm ERR! notarget It was specified as a dependency of 'src'
npm ERR! notarget

npm ERR! A complete log of this run can be found in:
npm ERR!     /opt/app-root/src/.npm/_logs/2019-10-28T11_30_27_657Z-debug.log
subprocess exited with status 1
subprocess exited with status 1
error: build error: error building at STEP "RUN /usr/libexec/s2i/assemble": exit
status 1
```

O processo de compilação falha e, portanto, nenhum aplicativo está em execução. O log de compilação indica que não há versão do pacote express que corresponda a uma especificação de versão 4.20.x.

- 6.3. Use o comando `oc get pods` para confirmar se o aplicativo não está implementado:

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
nodejs-dev-1-build   0/1     Error      0          2m
```

7. Atualize a versão da dependência express no arquivo `package.json` com um valor 4.x. Faça commit e envie as alterações por push para o repositório Git.
- 7.1. Edite o arquivo `package.json` no subdiretório `nodejs-app` e altere a versão da dependência express para 4.x. Salve o arquivo.

```
[student@workstation D0180-apps]$ cd nodejs-app
```

```
[student@workstation nodejs-app]$ sed -i s/4.20/4.x/ package.json
```

O arquivo contém o seguinte conteúdo:

```
[student@workstation nodejs-app]$ cat package.json
{
  "name": "nodejs-app",
  "version": "1.0.0",
  "description": "Hello World App",
  "main": "server.js",
  "author": "Red Hat Training",
  "license": "ASL",
  "dependencies": {
    "express": "4.x",
    "html-errors": "latest"
  }
}
```

- 7.2. Faça commit das alterações feitas e envie-as ao projeto.

Na janela de terminal, execute o seguinte comando para fazer commit e enviar as alterações:

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

8. Compile o aplicativo novamente. Verifique se o aplicativo é compilado sem erros.

- 8.1. Use o comando `oc start-build` para compilar o aplicativo novamente.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-2 started
```

- 8.2. Use o comando `oc logs` para monitorar os logs do processo de compilação:

```
[student@workstation nodejs-app]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
Pushing image ...image-registry.svc:5000/nodejs-app/nodejs-dev:latest ...
...output omitted...
Push successful
```

A compilação é bem-sucedida se uma imagem é transmitida ao registro interno do OpenShift.

9. Verifique se o aplicativo não está em execução devido a um erro de tempo de execução. Revise os logs e identifique o problema.

- 9.1. Use o comando `oc get pods` para verificar o status da implantação do pod do aplicativo. Em algum momento você verá que a primeira implantação do aplicativo tem o status `CrashLoopBackoff`.

```
[student@workstation nodejs-app]$ oc get pods
NAME          READY   STATUS        RESTARTS   AGE
nodejs-dev-1-build  0/1     Error        0          4m27s
nodejs-dev-1-deploy 0/1     Completed    0          28s
nodejs-dev-1-skf56  0/1     CrashLoopBackOff 2          25s
nodejs-dev-2-build  0/1     Completed    0          58s
```

- 9.2. Use o comando `oc logs -f deployment/nodejs-dev` para seguir os logs para a implantação do aplicativo:

```
[student@workstation nodejs-app]$ oc logs -f deployment/nodejs-dev
Environment:
  DEV_MODE=false
  NODE_ENV=production
  DEBUG_PORT=5858
...output omitted...

Error: Cannot find module 'http-error'
```

```
...output omitted...

npm info nodejs-app@1.0.0 Failed to exec start script
...output omitted...
npm ERR! code ELIFECYCLE
npm ERR! errno 1
npm ERR! nodejs-app@1.0.0 start: node `server.js`
npm ERR! Exit status 1
npm ERR!
npm ERR! Failed at the nodejs-app@1.0.0 start script.
npm ERR! This is probably not a problem with npm. There is likely additional
logging output above.
npm timing npm Completed in 159ms
...output omitted...
```

O log indica que o arquivo `server.js` tenta carregar um módulo chamado `http-error`. A variável `dependencies` no arquivo `packages` indica que o nome do módulo é `html-errors`, não `http-error`.

10. O log indica que o arquivo `server.js` tenta carregar um módulo chamado `http-error`, que não está disponível. Altere o módulo necessário na primeira linha de `server.js` para `html-errors`, que é um módulo válido no arquivo `packages`. Faça commit e envie por push as alterações do aplicativo para o repositório Git. Compile o aplicativo novamente. Depois que o aplicativo for compilado, verifique se ele está sendo executado.
 - 10.1. Corrija o erro de digitação do módulo na primeira linha do `server.js` de `http-error` para `html-errors`. Salve o arquivo.

```
[student@workstation nodejs-app]$ sed -i s/http-error/html-errors/ server.js
```

O arquivo contém o seguinte conteúdo:

```
[student@workstation nodejs-app]$ cat server.js
var createError = require('html-errors');

var express = require('express');
app = express();

app.get('/', function (req, res) {
  res.send('Hello World from pod: ' + process.env.HOSTNAME + '\n')
});

app.listen(8080, function () {
  console.log('Example app listening on port 8080!');
});
```

- 10.2. Faça commit das alterações feitas e envie-as ao projeto.

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

10.3. Use o comando `oc start-build` para compilar o aplicativo novamente.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-3 started
```

10.4. Use o comando `oc logs` para monitorar os logs do processo de compilação:

```
[student@workstation nodejs-app]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
Pushing image ...-image-registry.svc:5000/nodejs-app/nodejs-dev:latest ...
...output omitted...
Push successful
```

10.5. Use o comando `oc get pods -w` para monitorar a implantação de pods para o aplicativo `nodejs-dev`:

```
[student@workstation nodejs-app]$ oc get pods -w
NAME          READY   STATUS    RESTARTS   AGE
nodejs-dev-1-build  0/1     Error      0          11m
nodejs-dev-1-deploy 0/1     Completed   0          7m11s
nodejs-dev-2-9nds4  1/1     Running     0          56s
nodejs-dev-2-build  0/1     Completed   0          7m41s
nodejs-dev-2-deploy 0/1     Completed   0          61s
nodejs-dev-3-build  0/1     Completed   0          94s
```

Após uma terceira compilação, a segunda implantação resulta no status `Running`.

11. Crie uma rota para o aplicativo e teste o acesso a ele. Espere uma mensagem de erro. Revise os logs para identificar o erro.

11.1. Use o comando `oc expose` para criar uma rota para o aplicativo `nodejs-dev`:

```
[student@workstation nodejs-app]$ oc expose svc nodejs-dev
route.route.openshift.io/nodejs-dev exposed
```

11.2. Use o comando `oc get route` para recuperar o URL da rota `nodejs-dev`:

```
[student@workstation nodejs-app]$ oc get route
NAME        HOST/PORT
nodejs-dev  nodejs-dev-your_user-nodejs-app.wildcard_domain ...
```

11.3. Use o `curl` para acessar a rota. Espere que uma mensagem de erro seja exibida.

```
[student@workstation nodejs-app]$ curl \
> nodejs-dev-${RHT_OCP4_DEV_USER}-nodejs-app.${RHT_OCP4_WILDCARD_DOMAIN}
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Internal Server Error</pre>
</body>
</html>
```

11.4. Revise os logs para a implantação nodejs-dev:

```
[student@workstation nodejs-app]$ oc logs -f deployment/nodejs-dev
Environment:
  DEV_MODE=false
  NODE_ENV=production
  DEBUG_PORT=5858
Launching via npm...
npm info it worked if it ends with ok
npm info using npm@6.14.8
npm info using node@v12.19.1
npm info lifecycle nodejs-app@1.0.0~prestart: nodejs-app@1.0.0
npm info lifecycle nodejs-app@1.0.0~start: nodejs-app@1.0.0

Example app listening on port 8080!
TypeError: Cannot read property 'HOSTNAME' of undefined
...output omitted...
```

A seção correspondente do arquivo `server.js` é:

```
app.get('/', function (req, res) {
  res.send('Hello World from pod: ' + process.env.HOSTNAME + '\n')
});
```

Um objeto `process` no Node.js contém uma referência a um objeto `env`, não um objeto `environment`.

12. Substitua `process.environment` por `process.env` no arquivo `server.js` para corrigir o erro. Faça commit e envie as alterações do aplicativo por push para o repositório Git. Compile o aplicativo novamente. Quando o novo aplicativo for implantado, verifique se o aplicativo não gera erros ao acessar o URL do aplicativo.

12.1. Substitua `process.environment` por `process.env` no arquivo `server.js` para corrigir o erro.

```
[student@workstation nodejs-app]$ sed -i \
> s/process.environment/process.env/ server.js
```

O arquivo contém o seguinte conteúdo:

```
[student@workstation nodejs-app]$ cat server.js
var createError = require('html-errors');

var express = require('express');
app = express();

app.get('/', function (req, res) {
  res.send('Hello World from pod: ' + process.env.HOSTNAME + '\n')
});

app.listen(8080, function () {
  console.log('Example app listening on port 8080!');
});
```

12.2. Faça commit das alterações feitas e envie-as ao projeto.

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

12.3. Use o comando `oc start-build` para compilar o aplicativo novamente.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-4 started
```

12.4. Use o comando `oc logs` para monitorar os logs do processo de compilação:

```
[student@workstation nodejs-app]$ oc logs -f bc/nodejs-dev
Cloning "https://github.com/yourgituser/D0180-apps" ...
...output omitted...
Pushing image ...image-registry.svc:5000/nodejs-app/nodejs-dev:latest ...
...output omitted...
Push successful
```

12.5. Use o comando `oc get pods` para monitorar a implantação de pods para o aplicativo `nodejs-dev`:

```
[student@workstation nodejs-app]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
nodejs-dev-1-build  0/1     Error      0          21m
nodejs-dev-1-deploy 0/1     Completed   0          17m
nodejs-dev-2-build  0/1     Completed   0          17m
nodejs-dev-2-deploy 0/1     Completed   0          11m
nodejs-dev-3-build  0/1     Completed   0          11m
nodejs-dev-3-deploy 0/1     Completed   0          20s
nodejs-dev-3-wlpps  1/1     Running     0          17s
nodejs-dev-4-build  0/1     Completed   0          48s
```

Após uma quarta compilação, a terceira implantação tem o status Running.

- 12.6. Use o comando `curl` para testar o aplicativo. O aplicativo exibe uma mensagem `Hello World` contendo o nome do host do pod do aplicativo:

```
[student@workstation nodejs-app]$ curl \
> nodejs-dev-${RHT_OCP4_DEV_USER}-nodejs-app.${RHT_OCP4_WILDCARD_DOMAIN}
Hello World from pod: nodejs-dev-3-wlpps
```

Avaliação

Avalie seu trabalho executando o comando `lab troubleshoot-review grade` a partir da sua máquina `workstation`. Corrija as falhas relatadas e execute novamente o script até obter êxito.

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

Encerramento

Na `workstation`, execute o comando `lab troubleshoot-review finish` para concluir este laboratório.

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

Isso conclui o laboratório.

Sumário

Neste capítulo, você aprendeu que:

- Os aplicativos geralmente registram atividades, como eventos, avisos e erros, para auxiliar na análise do comportamento do aplicativo.
- Os aplicativos de contêineres devem imprimir dados de logs na saída padrão, em vez de em um arquivo, a fim de permitir acesso fácil aos logs.
- Para revisar os logs de um contêiner implantado localmente com o Podman, usa-se o comando `podman logs`.
- Usa-se o comando `oc logs` para acessar logs para objetos `BuildConfig` e `Deployment`, bem como pods individuais dentro de um projeto OpenShift.
- A opção `-f` permite monitorar a saída de log quase em tempo real para os comandos `podman logs` e `oc logs`.
- Use o comando `oc port-forward` para se conectar diretamente a uma porta em um pod de aplicativo. Você só deve aproveitar essa técnica em pods que não sejam de produção, porque interações podem alterar o comportamento do pod.

capítulo 9

Revisão abrangente

Meta

Revisar tarefas de *Red Hat OpenShift I: Containers & Kubernetes*

Objetivos

- Revisar tarefas de *Red Hat OpenShift I: Containers & Kubernetes*

Seções

- Revisão abrangente

Laboratório

- Revisão abrangente do *Introduction to Containers, Kubernetes, and Red Hat OpenShift*

Revisão abrangente

Objetivos

Depois de concluir esta seção, você deverá ter analisado e atualizado o conhecimento e as habilidades aprendidas em *Red Hat OpenShift I: Containers & Kubernetes*.

Revisão de Red Hat OpenShift I: Containers & Kubernetes

Para iniciar a revisão abrangente deste curso, você precisa estar familiarizado com os tópicos abordados em cada capítulo.

Você pode consultar as seções anteriores do material didático para estudo adicional.

Capítulo 1, Introdução à tecnologia de contêiner

Descrever como os aplicativos podem ser executados em contêineres orquestrados pelo Red Hat OpenShift Container Platform.

- Descrever a diferença entre os aplicativos de contêiner e implantações tradicionais.
- Descrever os conceitos básicos da arquitetura de contêineres.
- Descrever os benefícios da orquestração de aplicativos e da OpenShift Container Platform.

Capítulo 2, Criação de serviços conteinerizados

Provisionar um serviço usando tecnologia de contêineres.

- Criar um servidor de banco de dados a partir de uma imagem de contêiner.

Capítulo 3, Gerenciamento de contêineres

Usar imagens predefinidas de contêineres para criar e gerenciar serviços conteinerizados.

- Gerenciar o ciclo de vida de um contêiner da criação à exclusão.
- Salvar dados do aplicativo de contêiner com armazenamento persistente.
- Descrever como usar o encaminhamento de portas para acessar um contêiner.

Capítulo 4, Gerenciamento de imagens de contêiner

Gerenciar o ciclo de vida de uma imagem de contêiner da criação à exclusão.

- Pesquisar e fazer pull de imagens de registros remotos.
- Exportar, importar e gerenciar imagens de contêiner localmente e em um registro.

Capítulo 5, Criação de imagens personalizadas de contêiner

Projetar e codificar um Containerfile para criar uma imagem personalizada de contêiner.

capítulo 9 | Revisão abrangente

- Descrever as abordagens para criar imagens personalizadas de contêiner.
- Criar uma imagem de contêiner usando comandos comuns do Containerfile.

Capítulo 6, Implantação de aplicativos conteinerizados no OpenShift

Implantar aplicativos de contêiner único no OpenShift Container Platform.

- Descrever a arquitetura do Kubernetes e do Red Hat OpenShift Container Platform.
- Criar recursos básicos do Kubernetes
- Criar uma rota para um serviço.
- Compilar um aplicativo usando o recurso Source-to-Image do OpenShift Container Platform.
- Criar um aplicativo com o console da web do OpenShift.

Capítulo 7, Implantação de aplicativos com vários contêineres

Implantar aplicativos que são conteinerizados usando várias imagens de contêineres.

- Descrever as considerações para conteinerizar aplicativos com várias imagens de contêineres.
- Implantar um aplicativo de vários contêineres no OpenShift Platform.
- Implantar um aplicativo no OpenShift usando um template.

Capítulo 8, Solução de problemas de aplicativos conteinerizados

Solucionar os problemas de um aplicativo em contêineres implantado no OpenShift.

- Solucionar os problemas de compilação e implantação de um aplicativo no OpenShift.
- Implementar técnicas para solucionar problemas e depurar aplicativos em contêineres

► Laboratório Aberto

Conteinerização e implantação de um aplicativo de software

Nesta revisão, você conteinerizará um Nexus Server, compilará e testará usando o Podman e, então, implantará em um cluster do OpenShift.

Resultados

Você deverá ser capaz de:

- Gravar um Containerfile que conteinerize com êxito um servidor Nexus.
- Compilar uma imagem de contêiner do servidor Nexus e implantá-la usando o Podman.
- Implantar a imagem de contêiner do servidor Nexus em um cluster do OpenShift.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula* do Capítulo 1 antes de executar qualquer comando desta prática.

Execute o script de configuração para esta revisão abrangente.

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

Os arquivos do laboratório estão localizados no diretório /home/student/D0180/labs/comprehensive-review. Os arquivos da solução estão localizados no diretório /home/student/D0180/solutions/comprehensive-review.

Instruções

Use as seguintes etapas para criar e testar um servidor Nexus em contêiner tanto localmente quanto no OpenShift:

1. Crie uma imagem de contêiner que inicia uma instância de um servidor Nexus:
 - O diretório /home/student/D0180/labs/comprehensive-review/image contém arquivos para criar a imagem do contêiner. Execute o script get-nexus-bundle.sh para recuperar os arquivos do servidor Nexus.
 - Grave um Containerfile que conteinerize com êxito um servidor Nexus. O Containerfile deve estar localizado no diretório /home/student/D0180/labs/comprehensive-review/image. O Containerfile também deve:
 - Usar uma imagem de base de ubi8/ubi:8.5 e definir um mantenedor arbitrário.
 - Definir a variável ARG NEXUS_VERSION como 2.14.3-02 e defina a variável de ambiente NEXUS_HOME como /opt/nexus.
 - Instale o pacote java-1.8.0-openjdk-devel.

- Execute um comando para criar um usuário e grupo `nexus`. Eles têm uma UID e uma GID 1001. Altere as permissões do diretório `${NEXUS_HOME} /` para 775.
 - Descompactar o arquivo `nexus-2.14.3-02-bundle.tar.gz` e no diretório `${NEXUS_HOME} /`. Adicione o `nexus-start.sh` ao mesmo diretório.

Execute um comando, `ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2`, para criar um symlink no contêiner. Execute um comando para alterar recursivamente a propriedade do diretório pessoal do Nexus a `nexus:nexus`.
 - Faça com que o contêiner seja executado com o usuário `nexus` e defina o diretório em funcionamento como `/opt/nexus`:
 - Defina um ponto de montagem de volume para o diretório de contêineres `/opt/nexus/sonatype-work`. O servidor Nexus armazena dados nesse diretório.
 - Defina o comando do contêiner padrão como `nexus-start.sh`.

Há dois arquivos `*.snippet` no diretório do laboratório `/home/student/D0180/labs/comprehensive-review/image` que oferecem os comandos necessários para criar a conta `nexus` e instalar o Java. Use os arquivos para ajudá-lo a gravar o `Containerfile`.

 - Compile a imagem do contêiner com o nome `nexus`.
2. Compile e teste a imagem de contêiner usando o Podman com uma montagem de volume:
- Use o script `/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh` para iniciar um novo contêiner com uma montagem de volume.
 - Revise os logs do contêiner para verificar se o servidor foi iniciado e está em execução.
 - Teste o acesso ao serviço de contêiner usando o URL: `http://127.0.0.1:18081/nexus`.
 - Remova os contêineres de teste.
3. Implante a imagem de contêiner do servidor Nexus no cluster do OpenShift. Você deve:
- Marcar a imagem de contêiner do servidor Nexus como `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest` e enviá-la por push ao repositório público correspondente no quay.io.
 - Criar um projeto OpenShift com um nome `${RHT_OCP4_DEV_USER}-review`.
 - Editar `deploy/openshift/resources/nexus-deployment.yaml` e substitua `RHT_OCP4_QUAY_USER` pelo nome de usuário do Quay. Criar recursos do Kubernetes.
 - Criar uma rota para o serviço Nexus. Verifique se você pode acessar `http://nexus- ${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/da workstation`.

Avaliação

Depois de implantar a imagem de contêiner do servidor Nexus no cluster do OpenShift, verifique seu trabalho executando o script de avaliação do laboratório:

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

Encerramento

Na workstation, execute o comando `lab comprehensive-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

► Solução

Conteinerização e implantação de um aplicativo de software

Nesta revisão, você conteinerizará um Nexus Server, compilará e testará usando o Podman e, então, implantará em um cluster do OpenShift.

Resultados

Você deverá ser capaz de:

- Gravar um Containerfile que conteinerize com êxito um servidor Nexus.
- Compilar uma imagem de contêiner do servidor Nexus e implantá-la usando o Podman.
- Implantar a imagem de contêiner do servidor Nexus em um cluster do OpenShift.

Antes De Começar

Certifique-se de ter concluído a *Exercício Guiado: Configuração do ambiente de sala de aula do Capítulo 1* antes de executar qualquer comando desta prática.

Execute o script de configuração para esta revisão abrangente.

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

Os arquivos do laboratório estão localizados no diretório /home/student/D0180/labs/comprehensive-review. Os arquivos da solução estão localizados no diretório /home/student/D0180/solutions/comprehensive-review.

Instruções

Use as seguintes etapas para criar e testar um servidor Nexus em contêiner tanto localmente quanto no OpenShift:

1. Crie uma imagem de contêiner que inicia uma instância de um servidor Nexus:
 - O diretório /home/student/D0180/labs/comprehensive-review/image contém arquivos para criar a imagem do contêiner. Execute o script `get-nexus-bundle.sh` para recuperar os arquivos do servidor Nexus.
 - Grave um Containerfile que conteinerize com êxito um servidor Nexus. O Containerfile deve estar localizado no diretório /home/student/D0180/labs/comprehensive-review/image. O Containerfile também deve:
 - Usar uma imagem de base de `ubi8/ubi:8.5` e definir um mantenedor arbitrário.
 - Definir a variável ARG `NEXUS_VERSION` como `2.14.3-02` e defina a variável de ambiente `NEXUS_HOME` como `/opt/nexus`.
 - Instale o pacote `java-1.8.0-openjdk-devel`.

capítulo 9 | Revisão abrangente

- Execute um comando para criar um usuário e grupo nexus. Eles têm uma UID e uma GID 1001. Altere as permissões do diretório \${NEXUS_HOME} para 775.
- Descompactar o arquivo `nexus-2.14.3-02-bundle.tar.gz` no diretório \${NEXUS_HOME}/. Adicione o `nexus-start.sh` ao mesmo diretório.

Execute um comando, `ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2`, para criar um symlink no contêiner. Execute um comando para alterar recursivamente a propriedade do diretório pessoal do Nexus a `nexus:nexus`.

- Faça com que o contêiner seja executado com o usuário `nexus` e defina o diretório em funcionamento como `/opt/nexus`:
- Defina um ponto de montagem de volume para o diretório de contêineres `/opt/nexus/sonatype-work`. O servidor Nexus armazena dados nesse diretório.
- Defina o comando do contêiner padrão como `nexus-start.sh`.

Há dois arquivos `*.snippet` no diretório do laboratório `/home/student/D0180/labs/comprehensive-review/image` que oferecem os comandos necessários para criar a conta `nexus` e instalar o Java. Use os arquivos para ajudá-lo a gravar o Containerfile.

- Compile a imagem do contêiner com o nome `nexus`.

- 1.1. Execute o script `get-nexus-bundle.sh` para recuperar os arquivos do servidor Nexus.

```
[student@workstation ~]$ cd /home/student/D0180/labs/comprehensive-review/image
[student@workstation image]$ ./get-nexus-bundle.sh
#####
# 100.0%
Nexus bundle download successful
```

- 1.2. Grave um Containerfile que conteinerize com êxito um servidor Nexus. Acesse o diretório `/home/student/D0180/labs/comprehensive-review/image` e crie o Containerfile.
 - Especifique a imagem base a ser usada:

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

- Insira um nome arbitrário e um e-mail como mantenedor:

```
FROM ubi8/ubi:8.5
MAINTAINER username <username@example.com>
```

- Defina um argumento de build para `NEXUS_VERSION` e uma variável de ambiente para `NEXUS_HOME`:

```
FROM ubi8/ubi:8.5
MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus
```

- Instale o pacote `java-1.8.0-openjdk-devel` usando o comando `yum`.

```
FROM ubi8/ubi:8.5
MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus

RUN yum install -y --setopt=tsflags=nodocs java-1.8.0-openjdk-devel && \
    yum clean all -y
```

- Crie o diretório pessoal do servidor e conta e o grupo de serviço. Torne o diretório pessoal de propriedade da conta de serviço e altere as permissões para 775.

```
RUN groupadd -r nexus -f -g 1001 && \
useradd -u 1001 -r -g nexus -m -d ${NEXUS_HOME} -s /sbin/nologin \
-c "Nexus User" nexus && \
chown -R nexus:nexus ${NEXUS_HOME} && \
chmod -R 755 ${NEXUS_HOME}
```

- Faça com que o contêiner seja executado como o usuário `nexus`.

USER nexus

- Instale o software do servidor Nexus em `NEXUS_HOME` e adicione o script de inicialização. Observe que a diretiva ADD extrairá os arquivos do Nexus.

```
ADD nexus-${NEXUS_VERSION}-bundle.tar.gz ${NEXUS_HOME}
ADD nexus-start.sh ${NEXUS_HOME}/
```

- Crie o link simbólico `nexus2` apontando para o diretório do servidor Nexus. Altere recursivamente a propriedade do diretório `${NEXUS_HOME}` para `nexus:nexus`.

```
RUN ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} \
${NEXUS_HOME}/nexus2
```

- Torne `/opt/nexus` o diretório de trabalho atual:

WORKDIR \${NEXUS_HOME}

- Defina um ponto de montagem de volume para reter os dados persistentes do servidor Nexus:

```
VOLUME ["/opt/nexus/sonatype-work"]
```

- Defina a instrução CMD como o arquivo de script `nexus-start.sh`.

```
CMD ["sh", "nexus-start.sh"]
```

O Containerfile concluído deverá ficar assim:

```
FROM ubi8/ubi:8.5

MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus

RUN yum install -y --setopt=tsflags=nodocs java-1.8.0-openjdk-devel \
    && yum clean all -y

RUN groupadd -r nexus -f -g 1001 \
    && useradd -u 1001 -r -g nexus -m -d ${NEXUS_HOME} -s /sbin/nologin \
        -c "Nexus User" nexus \
    && chown -R nexus:nexus ${NEXUS_HOME} \
    && chmod -R 755 ${NEXUS_HOME}

USER nexus

ADD nexus-${NEXUS_VERSION}-bundle.tar.gz ${NEXUS_HOME}
ADD nexus-start.sh ${NEXUS_HOME}/

RUN ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2

WORKDIR ${NEXUS_HOME}

VOLUME ["/opt/nexus/sonatype-work"]

CMD ["sh", "nexus-start.sh"]
```

1.3. Compile a imagem do contêiner com o nome `nexus`.

```
[student@workstation image]$ podman build --layers=false -t nexus .
STEP 1: FROM ubi8/ubi:8.5
Getting image source signatures
...output omitted...
STEP 13: COMMIT ...output omitted...localhost/nexus:latest
...output omitted...
```

2. Compile e teste a imagem de contêiner usando o Podman com uma montagem de volume:

- Use o script `/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh` para iniciar um novo contêiner com uma montagem de volume.

- Revise os logs do contêiner para verificar se o servidor foi iniciado e está em execução.
 - Teste o acesso ao serviço de contêiner usando o URL: `http://127.0.0.1:18081/nexus`.
 - Remova os contêineres de teste.
- 2.1. Execute o script `run-persistent.sh`. Substitua o nome de contêiner como mostrado na saída do comando `podman ps`.

```
[student@workstation images]$ cd /home/student/D0180/labs/comprehensive-review
[student@workstation comprehensive-review]$ cd deploy/local
[student@workstation local]$ ./run-persistent.sh
80970007036bbb313d8eeb7621fada0ed3f0b4115529dc50da4dccef0da34533
```

- 2.2. Revise os logs do contêiner para verificar se o servidor foi iniciado e está em execução.

```
[student@workstation local]$ podman ps \
> --format="{{.ID}} {{.Names}} {{.Image}}"
81f480f21d47 inspiring_poincare localhost/nexus:latest
[student@workstation local]$ podman logs inspiring_poincare | grep JettyServer
...output omitted...
... INFO [jetty-main-1] ...jetty.JettyServer - Running
... INFO [main] ...jetty.JettyServer - Started
```

- 2.3. Use o comando `curl` para testar o contêiner usando a URL:
`http://127.0.0.1:18081/nexus`.

```
[student@workstation local]$ curl -v 127.0.0.1:18081/nexus/ 2>&1 \
> | grep -E 'HTTP|<title>' 
> GET /nexus/ HTTP/1.1
< HTTP/1.1 200 OK
<title>Nexus Repository Manager</title>
```

- 2.4. Remova os contêineres de teste.

```
[student@workstation local]$ podman rm -f inspiring_poincare
81f480f21d475af683b4b003ca6e002d37e6aaa581393d3f2f95a1a7b7eb768b
```

3. Implante a imagem de contêiner do servidor Nexus no cluster do OpenShift. Você deve:
- Marcar a imagem de contêiner do servidor Nexus como `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest` e enviá-la por push ao repositório público correspondente no quay.io.
 - Criar um projeto OpenShift com um nome `${RHT_OCP4_DEV_USER}-review`.
 - Editar `deploy/openshift/resources/nexus-deployment.yaml` e substitua `RHT_OCP4_QUAY_USER` pelo nome de usuário do Quay. Criar recursos do Kubernetes.
 - Criar uma rota para o serviço Nexus. Verifique se você pode acessar `http://nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/` da workstation.

capítulo 9 | Revisão abrangente

- 3.1. Faça login na sua conta do Quay.io.

```
[student@workstation local]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password: your_quay_password
Login Succeeded!
```

- 3.2. Publique a imagem de contêiner do servidor Nexus no registro quay.io.

```
[student@workstation local]$ podman push localhost/nexus:latest \
> quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

- 3.3. Os repositórios criados ao enviar imagens por push a quay.io são privados por padrão. Consulte a seção **Repositories Visibility** do Apêndice C para ler detalhes sobre como alterar a visibilidade do repositório.

- 3.4. Crie o projeto OpenShift:

```
[student@workstation local]$ cd ~/DO180/labs/comprehensive-review/deploy/openshift
[student@workstation openshift]$ oc login -u ${RHT_OCP4_DEV_USER} \
> -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation openshift]$ oc new-project ${RHT_OCP4_DEV_USER}-review
Now using project ...output omitted...
```

- 3.5. Substitua RHT_OCP4_QUAY_USER no arquivo de recursos pelo nome de usuário do Quay e crie os recursos do Kubernetes:

```
[student@workstation openshift]$ export RHT_OCP4_QUAY_USER
[student@workstation openshift]$ envsubst < resources/nexus-deployment.yaml \
> | oc create -f -
service/nexus created
persistentvolumeclaim/nexus created
deployment.apps/nexus created
[student@workstation openshift]$ oc get pods
NAME           READY   STATUS      RESTARTS   AGE
nexus-77c479bb4f-kscz7   0/1     ContainerCreating   0          11s
```

- 3.6. Exponha o serviço criando uma rota:

```
[student@workstation openshift]$ oc expose svc/nexus
route.route.openshift.io/nexus exposed.
[student@workstation openshift]$ oc get route -o yaml
apiVersion: v1
items:
- apiVersion: route.openshift.io/v1
  kind: Route
...output omitted...
```

```
spec:  
  host: nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}  
  ...output omitted...
```

- 3.7. Use um navegador para se conectar ao aplicativo web do servidor Nexus em `http://nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/`.

Avaliação

Depois de implantar a imagem de contêiner do servidor Nexus no cluster do OpenShift, verifique seu trabalho executando o script de avaliação do laboratório:

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

Encerramento

Na workstation, execute o comando `lab comprehensive-review finish` para concluir este laboratório.

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

Isso conclui o laboratório.

apêndice A

Implementação de arquitetura de microsserviços

Meta

Refatorar um aplicativo em microsserviços.

Objetivos

- Dividir um aplicativo entre vários contêineres para separar diferentes camadas e serviços.

Seções

- Implementação de arquitetura de microsserviços (com exercício orientado)

Implementação de arquiteturas de microsserviços

Objetivos

Depois de concluir esta seção, você deverá ser capaz de:

- Dividir um aplicativo entre vários contêineres para separar diferentes camadas e serviços.
- Descrever abordagens típicas para separar um aplicativo monolítico em várias unidades implantáveis.
- Descrever como separar o aplicativo `To Do List` em três contêineres que correspondem a suas camadas lógicas.

Benefícios da separação de um aplicativo monolítico em contêineres

A implantação tradicional de aplicativos normalmente tem muitas funções distintas empacotadas como uma única unidade de implantação ou um aplicativo monolítico. O desenvolvimento tradicional também pode implantar serviços de suporte, como bancos de dados e outros serviços de middleware, no mesmo servidor que o aplicativo. Embora os aplicativos monolíticos ainda possam ser implantados em um contêiner, muitas das vantagens de uma arquitetura de contêiner, como escalabilidade e agilidade, não são tão predominantes. A separação de monólitos exige consideração cuidadosa, e recomenda-se que, em aplicativos de microsserviços, cada microsserviço execute a funcionalidade mínima que pode ser executada isoladamente em cada contêiner.

Contêineres menores e a separação de um aplicativo e seus serviços de suporte em vários contêineres oferece muitas vantagens, como:

- Maior uso de hardware porque contêineres menores adaptam-se mais facilmente à capacidade disponível do host.
- Escalamento mais fácil porque as partes do aplicativo podem ser escaladas para oferecer suporte a uma carga de trabalho maior sem escalar outras partes do aplicativo.
- Upgrades mais fáceis porque os desenvolvedores podem atualizar partes do aplicativo sem afetar outras partes do mesmo aplicativo.

Duas formas comuns de separar um aplicativo são:

- Camadas: com base nas camadas arquitetônicas.
- Serviços: com base no recurso do aplicativo.

Divisão baseada em camadas

Uma maneira comum de os desenvolvedores organizarem os aplicativos é em camadas, com base na proximidade das funções com os usuários finais e com o armazenamento de dados. Um bom exemplo da arquitetura tradicional em três camadas é apresentação, lógica empresarial e persistência.

apêndice A | Implementação de arquitetura de microsserviços

Essa arquitetura lógica geralmente corresponde a uma arquitetura física de implantação, na qual a camada de apresentação seria implantada em um servidor web, a camada empresarial em um servidor de aplicativo e a camada de persistência em um servidor de banco de dados.

Separar um aplicativo em camadas permite que os desenvolvedores se especializem em tecnologias específicas com base nas camadas do aplicativo. Por exemplo, alguns desenvolvedores se concentram em aplicativos da web, enquanto outros preferem o desenvolvimento de bancos de dados. Outra vantagem é a habilidade de fornecer implementações alternativas de camadas com base em tecnologias diferentes. Por exemplo, criar um aplicativo para dispositivos móveis como outro front-end para um aplicativo existente. O aplicativo móvel poderia ser uma camada de apresentação alternativa, reutilizando as camadas empresarial e de persistência do aplicativo web original.

Normalmente, as camadas comercial e de apresentação de aplicativos menores são implantadas como uma unidade única. Por exemplo, no mesmo servidor web. Porém, conforme a carga aumenta, a camada de apresentação é movida para sua própria unidade de implantação para espalhar a carga. Os aplicativos menores podem, até mesmo, incorporar o banco de dados. Com frequência, os desenvolvedores compilam e implantam aplicativos mais exigentes dessa maneira monolítica.

Quando os desenvolvedores separam um aplicativo monolítico em camadas, eles geralmente aplicam várias alterações:

- Parâmetros de conexão para um banco de dados e outros serviços de middleware, como mensagens, eram codificados em endereços IP ou nomes de host fixos, geralmente `localhost`. Eles precisam ser parametrizados a fim de apontar para servidores externos que podem ser diferentes do desenvolvimento à produção.
- No caso dos aplicativos web, as chamadas do Ajax não podem ser feitas usando URLs relativos. Eles precisam usar um URL absoluto apontando para um nome de host DNS fixo público.
- Navegadores da web modernos rejeitam as chamadas do Ajax para servidores diferentes daquele que contém o script que faz o chamado, como medida de segurança. O aplicativo precisa ter permissões para o compartilhamento de recursos entre origens (CORS).

Depois que camadas de aplicativos são divididas para que possam ser executadas em diferentes servidores, não deverá haver problema para executá-las de diferentes contêineres.

Divisão baseada em serviços discretos

A maioria dos aplicativos complexos são compostos de vários serviços semi-independentes. Por exemplo, uma loja on-line teria catálogo de produtos, carrinho, pagamento, envio etc.

Quando um serviço específico em um aplicativo monolítico é reduzido, o dimensionamento do serviço para melhorar o desempenho implica no dimensionamento de todos os outros serviços de aplicativos constituintes. Se, no entanto, o serviço reduzido fizer parte de uma arquitetura de microsserviços, o serviço afetado será dimensionado independentemente dos outros serviços de aplicativo. A figura a seguir ilustra o dimensionamento de serviços tanto para arquitetura monolítica e baseada em microsserviço:

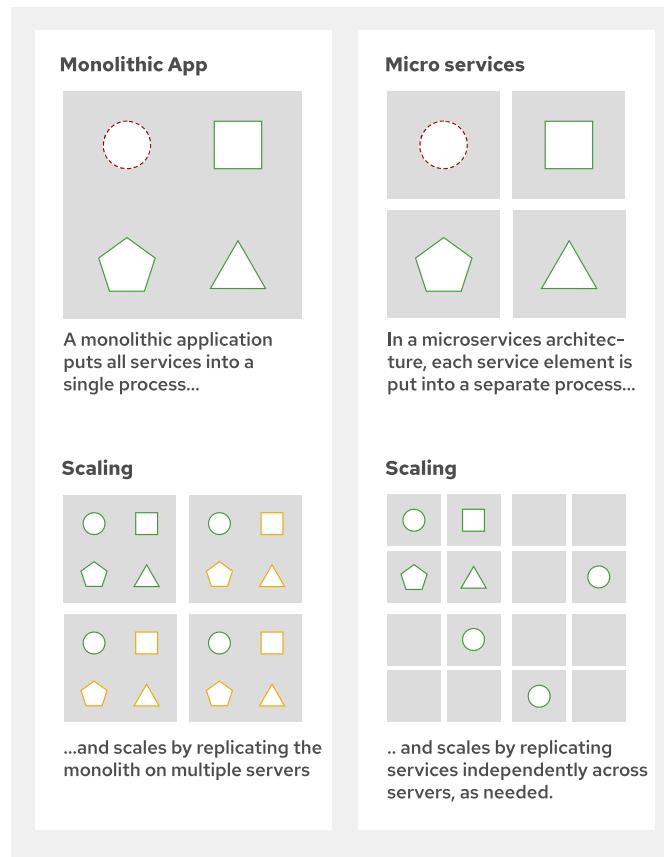


Figura A.1: Comparação do escalonamento de aplicativos em uma arquitetura monolítica versus uma arquitetura de microsserviços



nota

No diagrama anterior:

- Os números em linhas tracejadas em vermelho representam serviços que são usados em excesso.
- Os números em linhas sólidas amarelas representam serviços que são subutilizados.
- Os números em linhas sólidas verdes representam serviços que são usados da maneira correta.

Tanto as tradicionais arquiteturas orientadas a serviços (SOA) quanto e as recentes arquiteturas de microsserviços empacotam e implantam esses conjuntos de funções como unidades distintas. Isso permite que cada função definida seja desenvolvida por sua própria equipe, atualizada e dimensionada sem interromper outros conjuntos de funções (ou serviços). Preocupações entre funções, como autenticação, também podem ser empacotadas e implantadas como serviços que são consumidos por outras implementações de serviços.

Dividir cada preocupação em um servidor separado pode resultar em muitos aplicativos. Eles são logicamente arquitetados, empacotados e implantados como um pequeno número de unidades e, às vezes, como uma única unidade monolítica usando uma abordagem de serviços.

apêndice A | Implementação de arquitetura de microsserviços

Os contêineres permitem que arquiteturas baseadas em serviços sejam materializadas durante a implantação. Essa é a razão pela qual os microsserviços e contêineres geralmente se juntam. Porém, os contêineres por si só não são suficientes; eles precisam ser complementados por ferramentas de orquestração para gerenciar dependências entre serviços.

A arquitetura de microsserviços leva as arquiteturas baseadas em serviço ao extremo. Um serviço é tão pequeno quanto pode ser (sem interromper um conjunto de funções) e é implantado e gerenciado como uma unidade independente, em vez de parte de um aplicativo maior. Dessa forma, microsserviços existentes podem ser reutilizados para criar novos aplicativos.

Para separar um aplicativo em serviços, ele precisa do mesmo tipo de mudança da separação em camadas. Por exemplo, parametrizar parâmetros de conexão para bancos de dados e outros serviços middleware e lidar com problemas de segurança de navegadores da Web.

Refatoração do aplicativo To Do List

O aplicativo To Do List é um aplicativo simples com um único conjunto de funções. Por isso, separá-lo em serviços realmente não faz sentido. No entanto, refatorá-lo em camadas comerciais e de apresentação, ou seja, em um front-end e back-end para implantar em contêineres diferentes, representa o mesmo tipo de mudanças que a separação de um aplicativo comum em serviços.

A figura a seguir mostra o aplicativo To Do List implantado em três contêineres, um para cada camada:

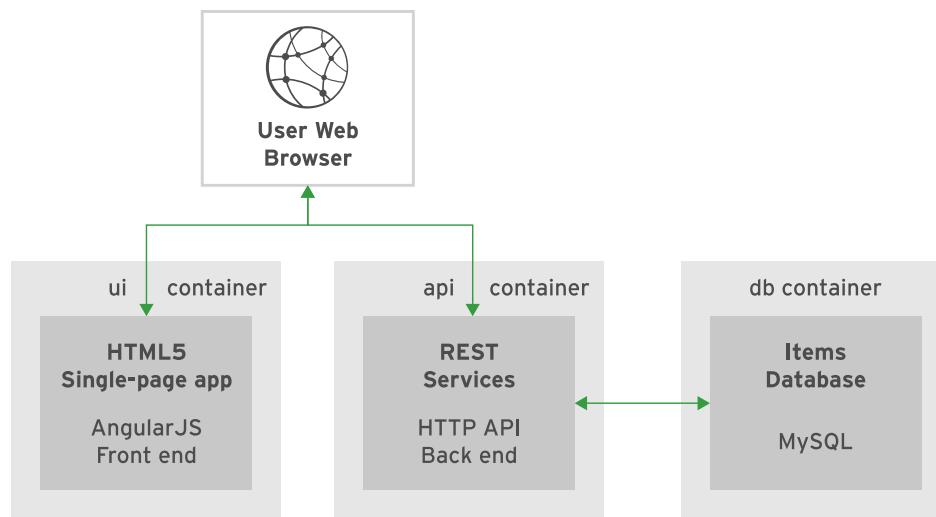


Figura A.2: Aplicativo To Do List separado em camadas e cada uma implantada como contêiner

Comparação do código-fonte do aplicativo monolítico com um refatorado; isso é uma visão geral das alterações.

- O JavaScript de front-end em `script/items.js` usa `workstation.lab.example.com` como nome do host para alcançar o back-end.
- O back-end usa variáveis de ambiente para obter parâmetros de conexão de banco de dados.
- O back-end precisa responder a solicitações usando o verbo de HTTP `OPTIONS` com cabeçalhos orientando o navegador da Web a aceitar solicitações vindas de diferentes domínios DNS usando CORS.

Outras versões do serviço de back-end podem ter mudanças similares. Cada linguagem de programação e estrutura REST tem sua própria sintaxe e seus próprios recursos.



Referências

Página do aplicativo monolítico na Wikipédia

https://en.wikipedia.org/wiki/Monolithic_application

Página do CORS na Wikipédia

https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

► Exercício Guiado

Refatoração do aplicativo To Do List

Neste laboratório, você refatorará o aplicativo To Do List em vários contêineres que estão vinculados uns aos outros, permitindo que o aplicativo de front-end HTML 5, a API REST Node.js e o servidor MySQL sejam executados em seus próprios contêineres.

Resultados

Você deverá ser capaz de refatorar um aplicativo monolítico em suas camadas e implantar cada camada como um microsserviço.

Antes De Começar

Execute o seguinte comando para configurar os diretórios de trabalho para o laboratório com os arquivos do aplicativo To Do List:

```
[student@workstation ~]$ lab appendix-microservices start
```

Instruções

► 1. Mova os arquivos HTML

A primeira etapa da refatoração do aplicativo To Do List é mover o código de front-end do aplicativo para seu próprio contêiner em execução. Essa etapa orienta você sobre como mover o aplicativo HTML e seus arquivos dependentes em seu próprio diretório para implantação em um servidor Apache sendo executado em um contêiner.

- 1.1. Mova os arquivos HTML e estáticos para o diretório `src/` a partir do aplicativo monolítico To Do List do Node.js:

```
[student@workstation ~]$ cd ~/D0180/labs/appendix-microservices/apps/html5/
[student@workstation html5]$ mv \
> ~/D0180/labs/appendix-microservices/apps/nodejs/todo/* \
> ~/D0180/labs/appendix-microservices/apps/html5/src/
```

- 1.2. O aplicativo de front-end atual interage com o serviço de API usando uma URL relativa. Como a API e o código de front-end serão agora executados em contêineres separados, o front-end precisa ser ajustado para apontar para a URL absoluta da API do aplicativo To Do List.

Abra o arquivo `/home/student/D0180/labs/appendix-microservices/apps/html5/src/script/item.js`. No final do arquivo, procure o seguinte método:

```
app.factory('itemService', function ($resource) {
  return $resource('api/items/:id');
});
```

Substitua o código pelo seguinte conteúdo:

```
app.factory('itemService', function ($resource) {
    return $resource('http://workstation.lab.example.com:30080/todo/api/
items/:id');
});
```

Certifique-se de que não há quebras de linha na nova URL, salve o arquivo e saia do editor.

► 2. Compile a imagem do HTML.

- 2.1. Faça login no Red Hat Container Catalog com sua conta da Red Hat.

```
[student@workstation html5]$ podman login registry.redhat.io
Username: your_username
Password: your_password
Login Succeeded!
```

- 2.2. Crie a imagem filha do Apache.

```
[student@workstation html5]$ cd ~/D0180/labs/appendix-microservices/deploy/html5
[student@workstation html5]$ ./build.sh
STEP 1: FROM registry.redhat.io/rhel8/httpd-24:1
Getting image source signatures
Copying blob 1b6a9fa02570 done
...output omitted...
Storing signatures
STEP 2: COPY ./src/ ${HOME}/
STEP 3: COMMIT do180/todo_frontend
dda10a4bf12ff987331e482e5cd87658abc5724da4b5b7d136874a9e471acf71
```

- 2.3. Verifique se a imagem foi compilada corretamente:

```
[student@workstation html5]$ podman images
REPOSITORY                                TAG      IMAGE ID      CREATED
SIZE
localhost/do180/todo_frontend              latest   dda10a4bf12f  About a minute ago
438 MB
registry.redhat.io/rhel8/httpd-24          1        adcf72f31a0b  13 days ago
438 MB
...
```

► 3. Modifique a API REST para se conectar a contêineres externos

- 3.1. Atualmente, a API REST usa valores codificados para se conectar ao banco de dados MySQL. Edite o arquivo /home/student/D0180/labs/appendix-microservices/apps/nodejs/models/db.js, que contém a configuração do banco de dados. Atualize o `dbname`, `username` e `password` valores para usar variáveis de ambiente. Além disso, atualize `params.host` para apontar para o nome do host que está em execução no contêiner MySQL e atualize o `params.port` para que reflita a porta redirecionada para o contêiner. Ambos os valores estão disponíveis como as variáveis de ambiente `MYSQL_SERVICE_HOST` e `MYSQL_SERVICE_PORT`, respectivamente. O conteúdo substituído deve ser assim:

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

**nota**

Este arquivo pode ser copiado e colado de `/home/student/D0180/solutions/appendix-microservices/apps/nodejs/models/db.js`.

- 3.2. Configure o back-end para lidar com o compartilhamento de recursos entre origens (CORS). Isso ocorre quando uma solicitação de recurso é feita de um domínio diferente daquele no qual a solicitação foi feita. Como a API precisa lidar com solicitações de um domínio DNS diferente (o aplicativo front-end), é necessário criar exceções de segurança para permitir que essas solicitações sejam bem-sucedidas. Faça as modificações a seguir ao aplicativo no editor de texto de sua preferência para lidar com CORS.

Adicione `"restify-cors-middleware": "1.1.1"` como uma nova dependência ao arquivo `package.json` localizado em `/home/student/D0180/labs/appendix-microservices/apps/nodejs/package.json`. Lembre-se de colocar uma vírgula no final da dependência anterior. Certifique-se de que o final do arquivo seja assim:

```
"sequelize": "5.21.1",
"mysql2": "2.0.0",
"restify-cors-middleware": "1.1.1"
}
```

Atualize o arquivo `app.js` localizado em `/home/student/D0180/labs/appendix-microservices/apps/nodejs/app.js` para configurar o uso do CORS. Exija o módulo `restify-cors-middleware` na segunda linha e, em seguida, atualize o conteúdo do arquivo para corresponder ao seguinte:

```
var restify = require('restify');
var corsMiddleware = require('restify-cors-middleware');
var controller = require('./controllers/items');
...output omitted...
var server = restify.createServer()
  .use(restify.plugins.fullResponse())
  .use(restify.plugins.queryParser())
  .use(restify.plugins.bodyParser());

const cors = corsMiddleware({ origins: ['*'] });

server.pre(cors.preflight);
```

```
server.use(cors.actual);

controller.context(server, '/todo/api', model);
```

As origens com o valor `["*"]` instruem o servidor a permitir qualquer domínio. Em um servidor de produção, esse valor geralmente seria uma matriz de domínios conhecida por solicitar acesso à API.

▶ 4. Criação da imagem da API REST

- 4.1. Compile a imagem filho da API REST, usando o comando a seguir. Essa imagem usa a imagem do Node.js image.

```
[student@workstation html5]$ cd ~/DO180/labs/appendix-microservices/deploy/nodejs
[student@workstation nodejs]$ ./build.sh
STEP 1: FROM quay.io/redhattraining/do180-todonodejs-12
Getting image source signatures
...output omitted...
STEP 6: CMD [ "/bin/bash", "-c", "./run.sh" ]
STEP 7: COMMIT do180/todonodejs
18f8d5dd8e25ed19a54750c8b96ae075adf437f5f0ba5ebbf7b9fe4b75526b3
```

- 4.2. Execute o comando `podman images` para verificar se todas as imagens necessárias foram compiladas com êxito:

```
[student@workstation nodejs]$ podman images
REPOSITORY                                TAG      IMAGE ID      CREATED
SIZE
localhost/do180/todonodejs                latest   18f8d5dd8e25  About a
minute ago     852 MB
localhost/do180/todo_frontend              latest   dda10a4bf12f  10 minutes
ago          438 MB
...output omitted...
```

▶ 5. Execução de contêineres

- 5.1. Use o script `run.sh` para executar os contêineres:

```
[student@workstation nodejs]$ cd linked/
[student@workstation linked]$ ./run.sh
• Creating database volume: OK
• Launching database: OK
• Importing database: OK
• Launching To Do application: OK
```

- 5.2. Execute o comando `podman ps` para confirmar que os três contêineres estão em execução:

```
[student@workstation linked]$ podman ps
... IMAGE ... PORTS NAMES
... localhost/do180/todo_frontend:latest ... 0.0.0.0:30000->80/tcp todo_frontend
... localhost/do180/todonodejs:latest ... 0.0.0.0:30080->30080/tcp todoapi
... registry.redhat.io/rhel8/mysql-80:1 ... 0.0.0.0:30306->3306/tcp mysql
```

► 6. Teste o aplicativo

- 6.1. Use o comando `curl` para verificar se a API REST do aplicativo To Do List está funcionando corretamente:

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

- 6.2. Abra o Firefox na `workstation` e navegue até 127.0.0.1:30000, onde você deverá ver o aplicativo To Do List.
- 6.3. Mude de volta para a pasta base do usuário.

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

Encerramento

Na `workstation`, execute o script `lab appendix-microservices finish` para concluir esse laboratório.

```
[student@workstation ~]$ lab appendix-microservices finish
```

Isso conclui o exercício orientado.

Sumário

Neste capítulo, você aprendeu que:

- A separação de um aplicativo monolítico em vários contêineres permite maior escalabilidade do aplicativo, facilita os upgrades e permite maior uso do hardware.
- As três camadas comuns para divisão lógica de um aplicativo são camada de apresentação, camada empresarial e camada de persistência.
- O compartilhamento de recursos entre origens (CORS) pode impedir o Ajax de chamar servidores diferentes daquele no qual foi feito o download das páginas. Certifique-se de fazer provisões para permitir o CORS de outros contêineres no aplicativo.
- As imagens de contêiner têm como objetivo serem imutáveis, mas as configurações podem ser transmitidas no momento da compilação da imagem ou criando armazenamento persistente para as configurações.

apêndice B

Criação de uma conta no GitHub

Meta

Descrever como criar uma conta do GitHub para os laboratórios do curso.

Criação de uma conta no GitHub

Objetivos

Depois de concluir esta seção, você deverá ser capaz de criar uma conta no GitHub e criar repositórios públicos do Git para os laboratórios do curso.

Criação de uma conta no GitHub

Você precisa de uma conta no GitHub para criar um ou mais repositórios git *públicos* para os laboratórios deste curso. Se você já tiver uma conta no GitHub, pode ignorar as etapas listadas neste apêndice.



Importante

Se você já tiver uma conta no GitHub, certifique-se de criar somente repositórios git *públicos* para os laboratórios deste curso. Os scripts e instruções de avaliação do laboratório exigem acesso não autenticado para clonar o repositório. Os repositórios devem estar acessíveis sem fornecer senhas, chaves SSH ou chaves GPG.

Para criar uma nova conta no GitHub, execute as seguintes etapas:

1. Navegue até <https://github.com> usando um navegador da web.
2. Insira os detalhes necessários e clique em **Create account**.

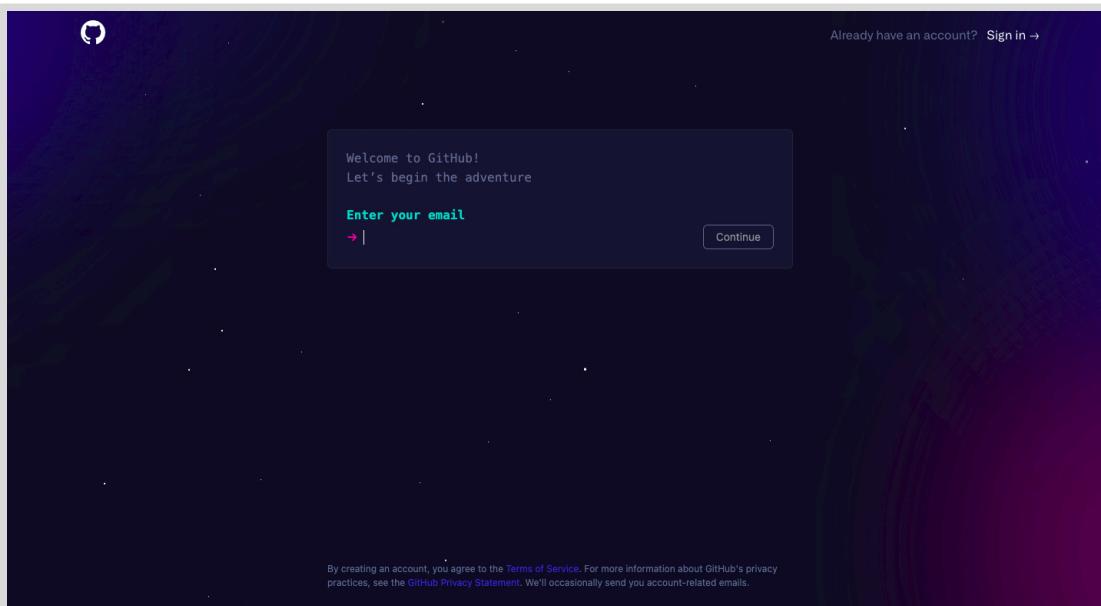


Figura B.1: Criação de uma conta no GitHub

3. Você receberá um e-mail com instruções sobre como ativar sua conta do GitHub. Verifique seu endereço de e-mail e, em seguida, entre no site do GitHub usando o nome de usuário e a senha que você forneceu durante a criação da conta.

4. Depois de fazer login no GitHub, você pode criar novos repositórios git clicando em **New** no painel **Repositories** à esquerda da página inicial do GitHub.

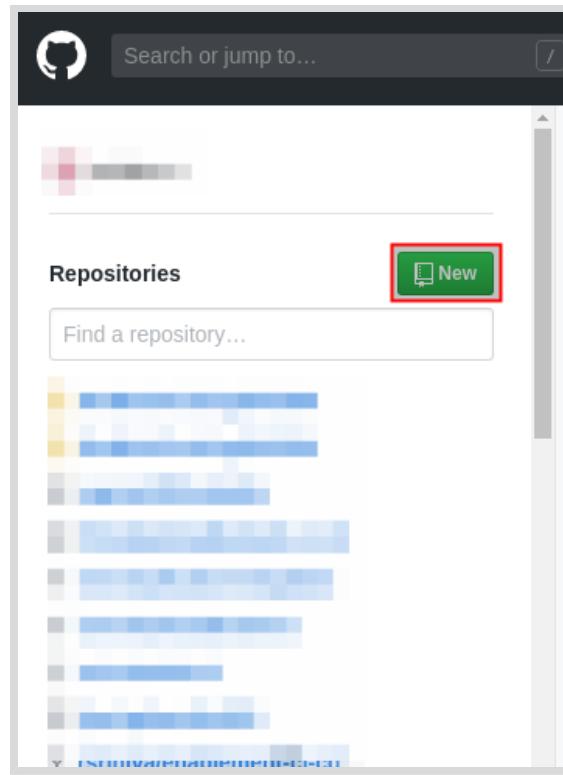


Figura B.2: Criação de um novo repositório git

Como alternativa, clique no ícone de mais (+) no canto superior direito (à direita do ícone de sino) e, em seguida, clique em **New repository**.

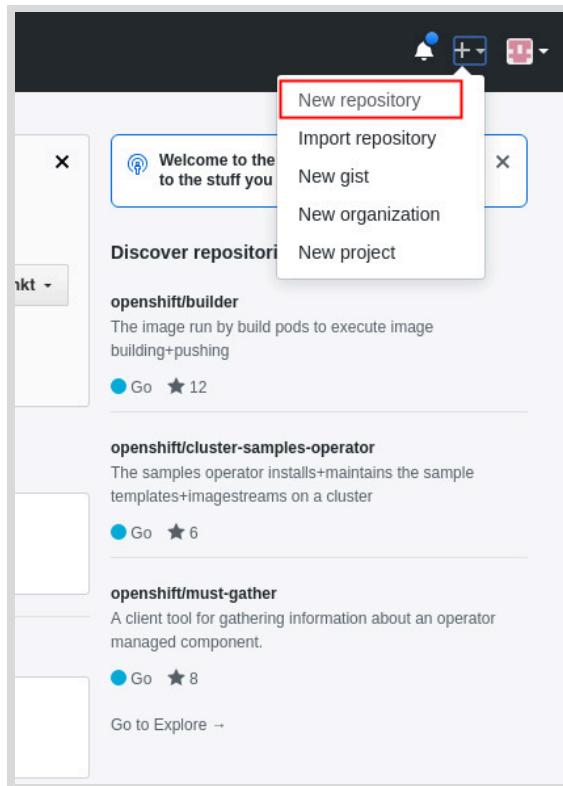


Figura B.3: Criação de um novo repositório git



Referências

Inscrição em uma nova conta do GitHub

<https://help.github.com/en/articles/signing-up-for-a-new-github-account>

apêndice C

Criação de uma conta no Quay

Meta

Descrever como criar uma conta do Quay para os laboratórios do curso.

Criação de uma conta no Quay

Objetivos

Depois de concluir esta seção, você deverá ser capaz de criar uma conta no Quay e criar repositórios públicos de imagem de contêiner para os laboratórios do curso.

Criação de uma conta no Quay

Você precisa de uma conta no Quay para criar um ou mais repositórios de imagem de contêiner *públicos* para os laboratórios deste curso. Se você já tiver uma conta no Quay, pode ignorar as etapas de criação de uma nova conta listadas neste apêndice.



Importante

Se você já tiver uma conta no Quay, certifique-se de criar somente repositórios de imagem de contêiner *públicos* para os laboratórios deste curso. Os scripts e instruções de avaliação do laboratório exigem acesso não autenticado para fazer pull das imagens de contêiner do repositório.

Para criara uma nova conta no Quay, execute as seguintes etapas:

1. Navegue até <https://quay.io> usando um navegador da web.
2. Clique em **Sign in** no canto superior direito (ao lado da barra de pesquisa).
3. Na página de **Sign in**, você pode fazer login usando suas credenciais da Red Hat (criadas no apêndice D).

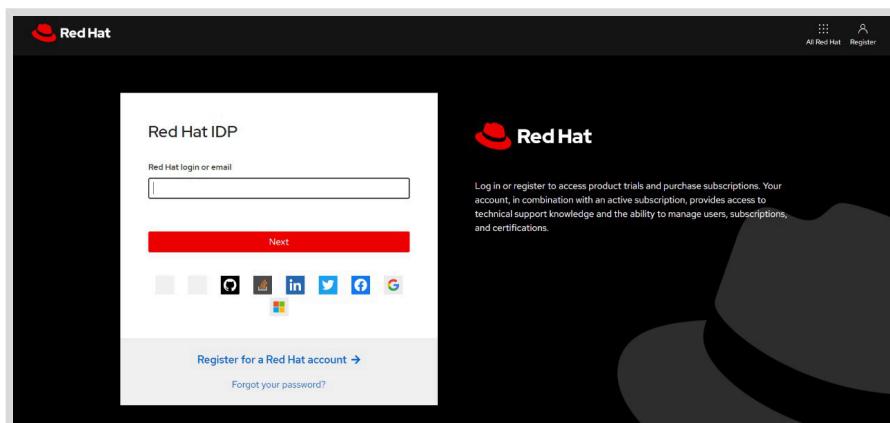


Figura C.1: Faça login usando as credenciais da Red Hat.

Depois de fazer login no Quay, você poderá criar novos repositórios de imagens clicando em **Create New Repository** na página **Repositories**.

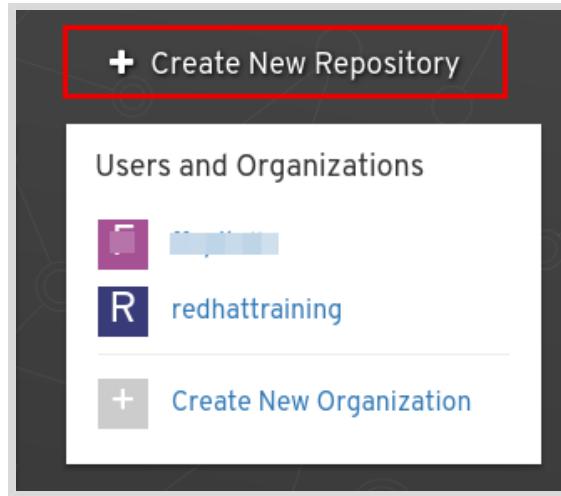


Figura C.2: Criação de um novo repositório de imagens

Como alternativa, clique no ícone de mais (+) no canto superior direito (à esquerda do ícone de sino) e, em seguida, clique em **New Repository**.

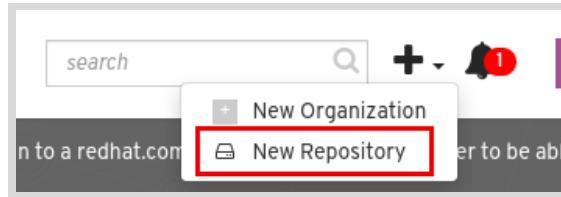


Figura C.3: Criação de um novo repositório de imagens

Trabalho com ferramentas CLI

Se você criou sua conta com a Red Hat, o Google ou o Github, precisará definir uma senha de conta para usar ferramentas da CLI, como Podman ou Docker.

1. Clique em <YOUR_USERNAME> no canto superior direito.
2. Clique em **Account Settings**.
3. Clique no link *Change password*.



Referências

Introdução ao Quay.io

<https://docs.quay.io/solution/getting-started.html>

Visibilidade de repositórios

Objetivos

Depois de concluir esta seção, você deverá ser capaz de controlar a visibilidade de repositórios no Quay.io.

Visibilidade de repositórios Quay.io

O Quay.io oferece a possibilidade de criar repositórios públicos e privados. Os repositórios públicos podem ser lidos por qualquer pessoa sem restrições, apesar de as permissões de gravação precisarem ser concedidas explicitamente. Os repositórios privados têm permissões de leitura e gravação restritas. No entanto, o número de repositórios privados em quay.io é limitado, dependendo do plano do namespace.

Visibilidade de repositórios padrão

Os repositórios criados ao enviar imagens por push a quay.io são privados por padrão. Para o OpenShift (ou qualquer outra ferramenta) buscar essas imagens, você pode configurar uma chave privada tanto no OpenShift quanto no Quay ou tornar o repositório público, de modo que nenhuma autenticação seja necessária. A configuração de chaves privadas está fora do escopo deste documento.

The screenshot shows the Red Hat Quay.io interface. At the top, there is a navigation bar with links for EXPLORE, APPLICATIONS, REPOSITORIES (which is highlighted in red), and TUTORIAL. Below the navigation bar, the page title is "Repositories". A section titled "Starred" is displayed, showing a message: "You haven't starred any repositories yet." It also includes a note: "Stars allow you to easily access your favorite repositories." Below this, there is a list of repositories under the heading "RHT_OCP4_QUAY_USER":

- do180-mysql-57-rhel7
- do180-todonodejs
- do180-quote-php
- nexus

Each repository entry has a star icon next to it, indicating it can be starred.

Atualização da visibilidade de repositórios

Para definir a visibilidade do repositório como pública, selecione o repositório apropriado em <https://quay.io/repository/> (faça login na sua conta se necessário) e abra a página **Settings**, clicando no ícone de engrenagem no canto inferior esquerdo. Role para baixo até a seção **Repository Visibility** e clique no botão **Make Public**.

The screenshot shows the settings page for a repository on Quay.io. At the top, there is a section titled "Trust Disabled" with a gear icon, stating "Signing is disabled on this repository." and a "Enable Trust" button. Below this is the "Events and Notifications" section, which says "No notifications have been setup for this repository." and "Click the 'Create Notification' button above to add a new notification for a repository event." The "Repository Visibility" section shows a lock icon and states "This Repository is currently **private**. Only users on the permissions list may view and interact with it." It includes a "Make Public" button. The final section is "Delete Repository", which contains a warning message: "Deleting a Repository **cannot be undone**. Here be dragons!" followed by a "Delete Repository" button.

Volte à lista de repositórios. O ícone de cadeado além do nome do repositório desapareceu, indicando que o repositório se tornou público.

apêndice D

Criação de uma conta da Red Hat

Meta

Descrever como criar uma conta da Red Hat para os laboratórios do curso.

Criação de uma conta da Red Hat

Objetivos

Depois de concluir esta seção, você deverá ser capaz de criar uma conta da Red Hat para acessar as imagens do Red Hat Container Catalog para os laboratórios no curso.

Criação de uma conta da Red Hat

Você precisa de uma conta da Red Hat para acessar as imagens do Red Hat Container Catalog. Se você já tiver uma conta da Red Hat, pode ignorar estas etapas.

Para criara uma nova conta da Red Hat , execute as seguintes etapas:

1. Navegue até <https://redhat.com> usando um navegador da web.
2. Clique em **Log in** no canto superior direito.
3. Clique em **Register now** no painel à direita.

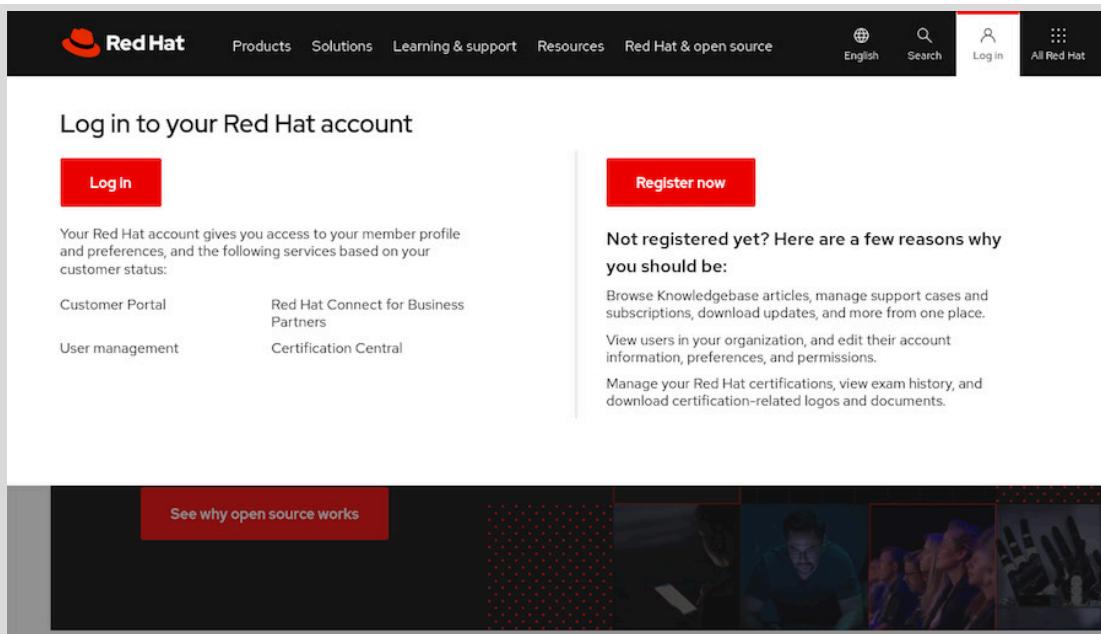
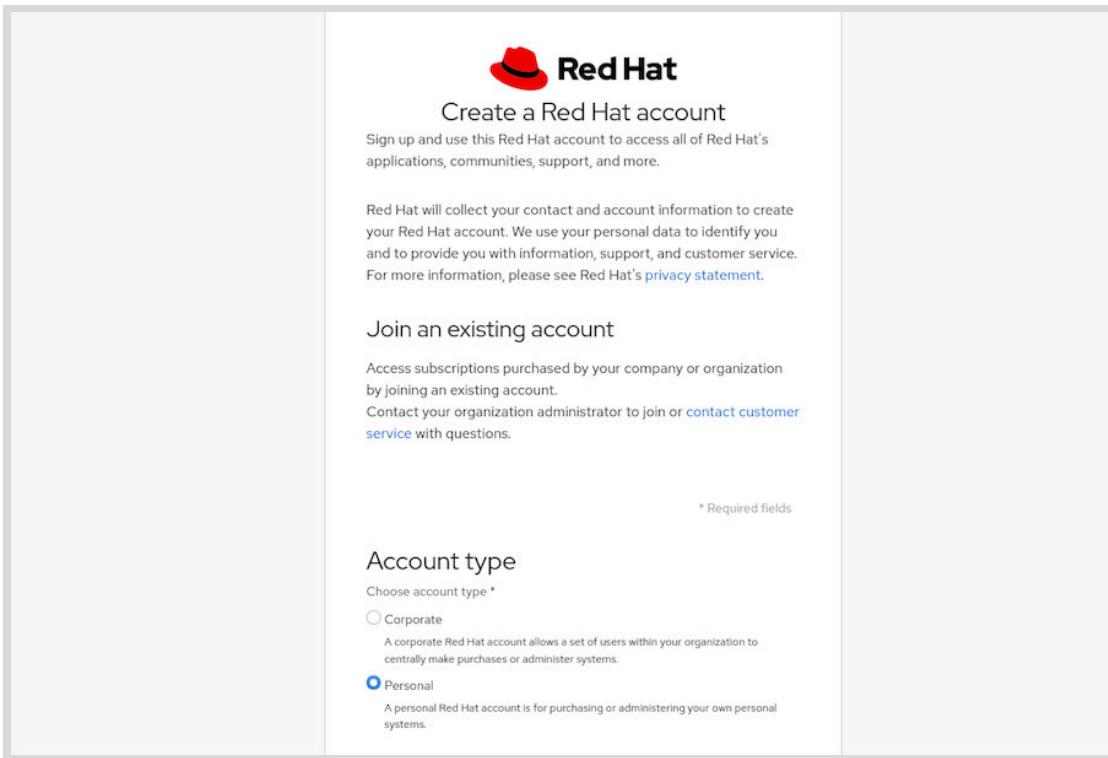


Figura D.1: Registre uma nova conta.

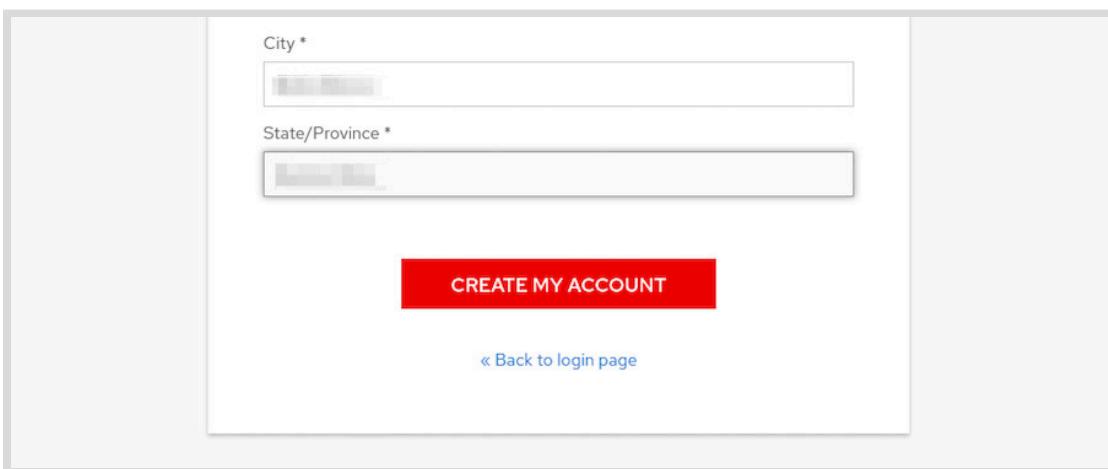
4. Defina Account type como Personal.



The screenshot shows the 'Create a Red Hat account' page. At the top is the Red Hat logo. Below it is the heading 'Create a Red Hat account'. A sub-instruction says 'Sign up and use this Red Hat account to access all of Red Hat's applications, communities, support, and more.' A note below states: 'Red Hat will collect your contact and account information to create your Red Hat account. We use your personal data to identify you and to provide you with information, support, and customer service. For more information, please see Red Hat's [privacy statement](#)'. A section titled 'Join an existing account' provides instructions for accessing company subscriptions or contacting an administrator. At the bottom right is a note: '* Required fields'.

Figura D.2: Defina o tipo de conta

5. Preencha o restante do formulário com suas informações pessoais. Nesse formulário, você também seleciona o nome de usuário e a senha dessa conta.
6. Clique em CREATE MY ACCOUNT.



The screenshot shows the 'CREATE MY ACCOUNT' form. It has two input fields: 'City *' and 'State/Province *', both with placeholder text. Below them is a large red 'CREATE MY ACCOUNT' button. At the bottom is a link '« Back to login page'.

Figura D.3: Preencha o formulário de informações pessoais

==

**Referências****Red Hat Customer Portal**<https://access.redhat.com/>

apêndice E

Comandos git úteis

Meta

Descrever comandos git úteis que são usados nos laboratórios deste curso.

Comandos git

Objetivos

Depois de concluir esta seção, você deverá ser capaz de reiniciar e refazer os exercícios neste curso. Você também deverá ser capaz de alternar de um exercício incompleto para executar outro e, posteriormente, continuar com o exercício anterior de onde você parou.

Trabalho com ramificações git

Este curso usa um repositório git hospedado no GitHub para armazenar o código-fonte do código do curso do aplicativo. No início do curso, você cria seu próprio fork desse repositório, que também é hospedada no GitHub.

Durante este curso, você trabalha com uma cópia local de sua ramificação, que você clona para a VM `workstation`. O termo origem se refere ao repositório remoto a partir do qual um repositório local é clonado.

Conforme você trabalha nos exercícios do curso, usa ramificações git separadas para cada exercício. Todas as alterações feitas no código-fonte ocorrem em uma nova ramificação que você cria somente para esse exercício. Nunca faça alterações na ramificação `master`.

Uma lista de cenários e os comandos git correspondentes que você pode usar para trabalhar com ramificações e recuperar para um estado em boas condições é exibida abaixo.

Como refazer um exercício do zero

Para refazer um exercício do zero depois de concluí-lo, execute as seguintes etapas:

1. Você confirma e envia por push todas as alterações em sua ramificação local como parte da realização do exercício. Você concluiu o exercício executando o subcomando `finish` para limpar todos os recursos:

```
[student@workstation ~]$ lab your-exercise finish
```

2. Altere para seu clone local do repositório D0180-apps e alterne para o branch `master`:

```
[student@workstation ~]$ cd ~/D0180-apps  
[student@workstation D0180-apps]$ git checkout master
```

3. Exclua sua ramificação local:

```
[student@workstation D0180-apps]$ git branch -d your-branch
```

4. Exclua o branch remoto de sua conta pessoal do GitHub:

```
[student@workstation D0180-apps]$ git push origin --delete your-branch
```

5. Use o subcomando `start` para reiniciar o exercício:

apêndice E | Comandos git úteis

```
[student@workstation D0180-apps]$ cd ~  
[student@workstation ~]$ lab your-exercise start
```

Como abandonar um exercício parcialmente concluído e reiniciá-lo do zero

É possível que você se depare com uma situação em que concluiu parcialmente algumas etapas no exercício e deseja abandonar a tentativa atual e reiniciá-la do zero. Execute estas etapas:

1. Execute o subcomando `finish` do exercício para limpar todos os recursos.

```
[student@workstation ~]$ lab your-exercise finish
```

2. Digite seu clone local do repositório D0180-apps e descarte todas as alterações pendentes no branch atual usando `git stash`:

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

3. Alterne para a ramificação `master` do seu repositório local:

```
[student@workstation D0180-apps]$ git checkout master
```

4. Exclua seu branch local:

```
[student@workstation D0180-apps]$ git branch -d your-branch
```

5. Exclua o branch remoto de sua conta pessoal do GitHub:

```
[student@workstation D0180-apps]$ git push origin --delete your-branch
```

6. Agora você pode reiniciar o exercício executando o subcomando `start`:

```
[student@workstation D0180-apps]$ cd ~  
[student@workstation ~]$ lab your-exercise start
```

Alternar para um exercício diferente de um exercício incompleto

Talvez você se depare com uma situação em que concluiu parcialmente algumas etapas em um exercício, mas deseja alternar para um exercício diferente e revisitar o exercício atual posteriormente.

Evite deixar muitos exercícios incompletos para revisitar mais tarde. Esses exercícios vinculam recursos de nuvem, e você pode usar sua cota atribuída no provedor de nuvem e no cluster do OpenShift que você compartilha com outros alunos. Se você achar que pode demorar até que consiga voltar ao exercício atual, considere abandoná-lo e reiniciá-lo do zero mais tarde.

Se preferir interromper o exercício atual e trabalhar no próximo, execute as seguintes etapas:

apêndice E | Comandos git úteis

1. Confirme todas as alterações pendentes em seu repositório local e envie-as para sua conta pessoal do GitHub. É recomendável registrar a etapa em que você parou o exercício:

```
[student@workstation ~]$ cd ~/D0180-apps  
[student@workstation D0180-apps]$ git commit -a -m "Paused at step X.Y"  
[student@workstation D0180-apps]$ git push
```

2. Não execute o comando **finish** do exercício original. Isso é importante para deixar seus projetos existentes do OpenShift inalterados, para que você possa continuar mais tarde.
3. Inicie o próximo exercício executando seu subcomando **start**:

```
[student@workstation ~]$ lab your-exercise start
```

4. O próximo exercício alterna para a ramificação **master** e, como alternativa, cria uma nova ramificação para suas alterações. Isso significa que as alterações feitas no exercício original na ramificação original são deixadas inalteradas.
5. Posteriormente, depois que você concluir o exercício seguinte e quiser voltar ao exercício original, alterne para sua ramificação:

```
[student@workstation ~]$ git checkout original-branch
```

Em seguida, você pode continuar com o exercício original na etapa em que parou.



Referências

Página do man da ramificação git

<https://git-scm.com/docs/git-branch>

O que é uma ramificação git?

<https://git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell>

Git Tools - Stashing

<https://git-scm.com/book/en/v2/Git-Tools-Stashing-and-Cleaning>