



Red hat OpenShift I: Container & Kubernetes



OCP 4.6 DO180
Red hat OpenShift I: Container & Kubernetes
Ausgabe 120210607
Veröffentlicht 20210607

Autoren: Zach Guterman, Dan Kolepp, Eduardo Ramirez Ronco,
Jordi Sola Alaball, Richard Allred, Michael Jarrett, Harpal Singh,
Federico Capitalle, Maria Fernanda Ordóñez Casado
Editor: 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 and/or its affiliates.

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

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

Node.js® is an official 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 OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation or the OpenStack community.

All other trademarks are the property of their respective owners.

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

Dokumentkonventionen	vii
Einführung	ix
DO180: Red Hat OpenShift I: Container & Kubernetes	ix
Informationen zur Kursumgebung	xi
Landessprachliche Anpassung	xv
.....	xx
1. Introducing Container Technology	1
Überblick über die Container-Technologie	2
Quiz: Überblick über die Container-Technologie	5
Überblick über die Container-Architektur	9
Quiz: Überblick über die Container-Architektur	12
Überblick über Kubernetes und OpenShift	14
Quiz: Beschreibung von Kubernetes und OpenShift	17
Angeleitete Übung: Konfigurieren der Kursumgebung	19
Zusammenfassung	29
2. Erstellen von containerisierten Services	31
Bereitstellen von containerisierten Services	32
Angeleitete Übung: Erstellen einer MySQL-Datenbankinstanz	38
Verwenden von rootless-Containern	41
Angeleitete Übung: Erkunden von root- und rootless-Containern	44
Praktische Übung: Erstellen von containerisierten Services	46
Zusammenfassung	50
3. Verwalten von Containern	51
Verwalten des Lebenszyklus von Containern	52
Angeleitete Übung: Verwalten von MySQL-Containern	60
Anhängen von persistentem Storage	64
Angeleitete Übung: MySQL-Container mit persistenter Datenbank erstellen	68
Zugreifen auf Container	71
Angeleitete Übung: Laden der Datenbank	73
Praktische Übung: Verwalten von Containern	76
Zusammenfassung	85
4. Verwalten von Container-Images	87
Zugreifen auf Registries	88
Quiz: Verwenden von Registries	95
Ändern von Container-Images	99
Angeleitete Übung: Erstellen eines benutzerdefinierten Apache-Container-Images	105
Praktische Übung: Verwalten von Images	110
Zusammenfassung	119
5. Erstellen benutzerdefinierter Container-Images	121
Konzipieren von benutzerdefinierten Container-Images	122
Quiz: Entwurfsansätze für Container-Images	127
Erstellen benutzerdefinierter Container-Images mit Containerfiles	129
Angeleitete Übung: Erstellen eines Apache-Basis-Container-Images	135
Praktische Übung: Erstellen benutzerdefinierter Container-Images	139
Zusammenfassung	146
6. Bereitstellen containerisierter Anwendungen in OpenShift	147
Beschreiben der Kubernetes- und OpenShift-Architektur	148
Quiz: Beschreibung von Kubernetes und OpenShift	155
Erstellen von Kubernetes-Ressourcen	159
Angeleitete Übung: Bereitstellen eines Datenbankservers auf OpenShift	172
Erstellen von Routen	177

Angeleitete Übung: Bereitstellen eines Service als Route	181
Erstellen von Anwendungen mit Source-to-Image	186
Angeleitete Übung: Erstellen einer containerisierten Anwendung mit Source-to-Image	196
Erstellung von Anwendungen mit der OpenShift-Webkonsole	202
Angeleitete Übung: Erstellen von Anwendungen mit der Web Console	208
Praktische Übung: Bereitstellen containerisierter Anwendungen in OpenShift	224
Zusammenfassung	228
7. Bereitstellen von Anwendungen mit mehreren Containern	229
Überlegungen zu Anwendungen mit mehreren Containern	230
Angeleitete Übung: Bereitstellen von Webanwendung und MySQL in Linux-Containern ...	235
Bereitstellen einer Anwendung mit mehreren Containern auf OpenShift	240
Angeleitete Übung: Erstellen einer Anwendung in OpenShift	242
Bereitstellen einer Anwendung mit mehreren Containern in OpenShift mithilfe einer Vorlage	246
Angeleitete Übung: Erstellen einer Anwendung mit einer Vorlage	253
Praktische Übung: Bereitstellen von Anwendungen mit mehreren Containern	257
Zusammenfassung	264
8. Fehlerbehebung bei containerisierten Anwendungen	265
Fehlerbehebung bei S2I-Builds und -Bereitstellungen	266
Angeleitete Übung: Fehlerbehebung bei OpenShift-Builds	272
Fehlerbehebung bei containerisierten Anwendungen	280
Angeleitete Übung: Konfigurieren von Apache-Containerprotokollen für das Debugging	287
Praktische Übung: Fehlerbehebung bei containerisierten Anwendungen	290
Zusammenfassung	301
9. Ausführliche Wiederholung	303
Ausführliche Wiederholung	304
Praktische Übung: Containerisieren und Bereitstellen einer Softwareanwendung	306
A. Implementieren der Microservices-Architektur	317
Implementieren der Microservices-Architektur	318
Angeleitete Übung: Refactoring der Anwendung „To Do List“	323
Zusammenfassung	328
B. Erstellen eines GitHub-Benutzerkontos	329
Erstellen eines GitHub-Benutzerkontos	330
C. Erstellen eines Quay-Benutzerkontos	333
Erstellen eines Quay-Benutzerkontos	334
Sichtbarkeit von Repositorys	337
D. Erstellen eines Red Hat-Benutzerkontos	341
Erstellen eines Red Hat-Benutzerkontos	342
E. Nützliche Git-Befehle	345
Git-Befehle	346

Dokumentkonventionen



Literaturhinweise

„Verweise“ geben an, wo Sie weitere Informationen zu einem Thema in externen Dokumentationen finden können.



Anmerkung

„Hinweise“ sind Tipps, Tastenkombinationen oder alternative Ansätze für die vorliegende Aufgabe. Wenn Sie einen Hinweis ignorieren, hat dies normalerweise keine negativen Konsequenzen. Allerdings können Hinweise helfen, einen Vorgang zu optimieren.



Wichtig

In den Feldern „Wichtig“ werden Details hervorgehoben, die andernfalls leicht übersehen werden könnten: Konfigurationsänderungen, die nur die aktuelle Sitzung betreffen, oder Services, die neu gestartet werden müssen, bevor ein Update angewendet werden kann. Wenn Sie ein Feld mit der Bezeichnung „Wichtig“ ignorieren, kommt es zwar nicht zu Datenverlusten, aber es kann zu Irritationen und Frustration führen.



Warnung

„Warnungen“ dürfen nicht ignoriert werden. Ignorierte Warnungen führen mit großer Wahrscheinlichkeit zu einem Datenverlust.

Einführung

DO180: Red Hat OpenShift I: Container & Kubernetes

Der praxisorientierte Kurs „DO180: Red Hat OpenShift I: Container & Kubernetes“ soll Teilnehmern vermitteln, wie Container mithilfe von Podman, Kubernetes und Red Hat OpenShift Container Platform erstellt, bereitgestellt und verwaltet werden.

Die kontinuierliche Integration (Continuous Integration, CI) und kontinuierliche Bereitstellung (Continuous Deployment, CD) zählen zu den wichtigsten Anliegen der DevOps-Bewegung. Container wurden zu einer wichtigen Technologie für die Konfiguration und Bereitstellung von Anwendungen und Microservices. Red Hat OpenShift Container Platform ist eine Implementierung des Containerorchestrierungssystems Kubernetes.

Lerninhalte

- Demonstrieren von Kenntnissen hinsichtlich des Container-Ökosystems.
- Verwalten von Linux-Containern mit Podman.
- Bereitstellen von Containern auf einem Kubernetes-Cluster mithilfe von OpenShift Container Platform.
- Demonstrieren des grundlegenden Container-Designs und der Fähigkeit, Container-Images zu erstellen.
- Implementieren einer containerbasierten Architektur mit dem Wissen in Bezug auf Container, Kubernetes und OpenShift.

Zielgruppe

- Systemadministratoren
- Entwickler
- IT-Führungskräfte und Infrastrukturarchitekten

Voraussetzungen Die Teilnehmer sollten mindestens eine der folgenden Voraussetzungen erfüllen:

- Sie sollten eine Linux-Terminalsitzung verwenden und Betriebssystembefehle ausführen können. Eine RHCSA-Zertifizierung wird empfohlen, ist jedoch nicht zwingend erforderlich.
- Sie sollten über Erfahrung mit Webanwendungsarchitekturen und den entsprechenden Technologien verfügen.

Informationen zur Kursumgebung

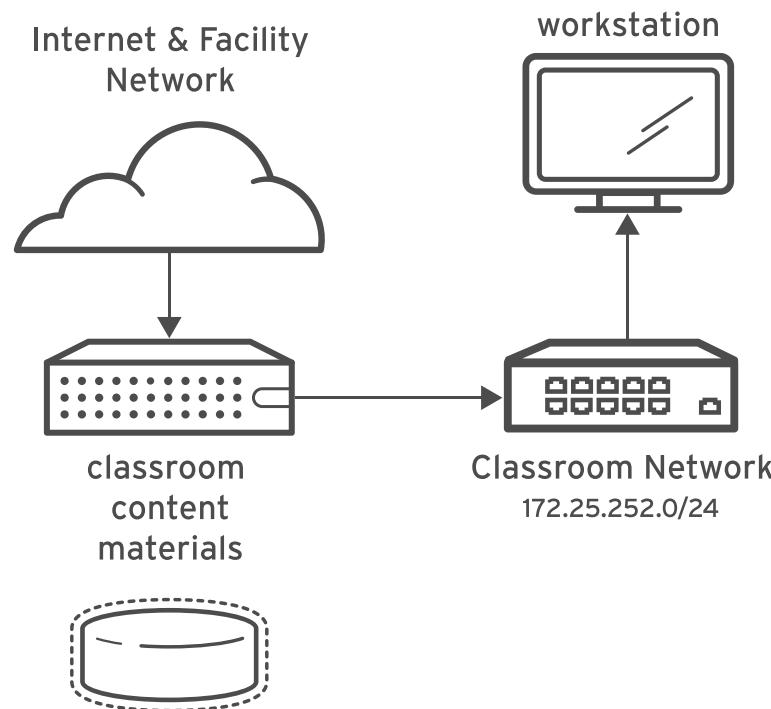


Abbildung 0.1: Kursumgebung

In diesem Kurs wird **workstation** als primäres Computersystem für praktische Übungen verwendet. Das ist ein virtueller Rechner (VM) mit dem Namen `workstation.lab.example.com`.

Alle Kursteilnehmer-Rechner verfügen über das standardmäßige Benutzerkonto `student` mit dem Passwort `student`. Das `root`-Passwort für alle Kursteilnehmer-Systeme lautet `redhat`.

Kursraum-Rechner

Rechnername	IP-Adressen	Rolle
<code>workstation.lab.example.com</code>	172.25.250.9	Grafische Workstation für die Systemadministration

Rechnername	IP-Adressen	Rolle
classroom.example.com	172.25.254.254	Router, der das Classroom Network mit dem Internet verbindet
bastion.lab.example.com	172.25.252.1	Router, der das Student Network mit dem Classroom Network verbindet.

Im Kursraum bieten verschiedene Systeme Unterstützung. Die beiden Server `content.example.com` und `materials.example.com` dienen als Quelle für Software- und Übungsmaterialien für praktische Übungen. Informationen zur Verwendung dieser Server finden Sie in der Anleitung der jeweiligen Übungen.

Die Kursteilnehmer greifen über den Rechner `workstation` auf einen gemeinsam genutzten OpenShift-Cluster zu, der extern gehostet wird. Die Kursteilnehmer verfügen über keine Administratorberechtigungen für den Cluster, dies ist jedoch nicht erforderlich, um den DO180-Inhalt abzuschließen.

Den Kursteilnehmern wird ein Benutzerkonto in einem freigegebenen OpenShift 4-Cluster bereitgestellt, wenn sie ihre Umgebungen in der Red Hat Online Learning-Schnittstelle bereitstellen. Clusterinformationen wie API-Endpunkt und Cluster-ID sowie ihr Benutzername und Kennwort werden ihnen bei der Bereitstellung ihrer Umgebung angezeigt.

Die Kursteilnehmer haben auch Zugriff auf einen MySQL- und einen Nexus-Server, der entweder vom OpenShift-Cluster oder von einem externen Anbieter gehostet wird. Praktische Aktivitäten in diesem Kurs enthalten Anweisungen für den Zugriff auf diese Server, sofern erforderlich.

Zum Durchführen der praktischen Aktivitäten in DO180 benötigen die Kursteilnehmer persönliche Benutzerkonten bei den zwei öffentlichen, kostenlosen Internetservices GitHub und Quay.io. Die Kursteilnehmer müssen diese Benutzerkonten einrichten, wenn sie noch nicht über sie verfügen (siehe Anhang), und ihren Zugriff überprüfen, indem sie sich bei diesen Services anmelden, bevor sie den Kurs beginnen.

Kursteilnehmern werden Remote-Computer in einem Red Hat Online Learning-Kursraum zugewiesen. Der Zugriff darauf erfolgt über eine unter `rol.redhat.com` [<http://rol.redhat.com>] gehostete Webanwendung. Kursteilnehmer sollten sich mithilfe ihrer Anmelddaten für das Red Hat Customer Portal auf dieser Website anmelden.

Steuern der virtuellen Rechner

Die virtuellen Rechner in Ihrer Kursumgebung werden über eine Webseite gesteuert. Der Status jedes virtuellen Rechners in der Kursumgebung wird auf der unter der Registerkarte Lab Environment befindlichen Seite angezeigt.

Rechnerstatus

VM-Status	Beschreibung
active	Der virtuelle Rechner wird ausgeführt und ist verfügbar (oder wird es bald sein, falls er noch bootet).
stopped	Der virtuelle Rechner ist vollständig heruntergefahren.
building	Der Ersterstellung des virtuellen Rechners wird durchgeführt.

Abhängig vom Status eines Rechners steht eine Auswahl der folgenden Aktionen zur Verfügung.

Aktionen für Kursumgebung/Rechner

Schaltfläche oder Aktion	Beschreibung
CREATE	Erstellt die ROL-Kursumgebung. Hiermit werden sämtliche für den Kursraum erforderlichen virtuellen Rechner erstellt und gestartet. Dies dauert ggf. mehrere Minuten.
DELETE	Entfernen Sie den ROL-Kursraum. Hiermit werden alle virtuellen Rechner im Kursraum entfernt. Achtung: Sämtliche auf den Disks gespeicherte Arbeit geht verloren.
START	Starten Sie alle virtuellen Rechner im Kursraum.
OPEN CONSOLE	Öffnet eine neue Registerkarte im Browser und stellt eine Verbindung zwischen Konsole und virtuellem Rechner her. Kursteilnehmer können sich direkt beim virtuellen Rechner anmelden und Befehle ausführen. In den meisten Fällen sollten sich die Kursteilnehmer beim virtuellen Rechner workstation anmelden und ssh verwenden, um mit anderen virtuellen Rechnern eine Verbindung herzustellen.
Starten	Starten Sie den virtuellen Rechner (d. h. schalten Sie ihn ein).
Herunterfahren	Fahren Sie den virtuellen Rechner ordnungsgemäß herunter, damit die Disk-Inhalte nicht verloren gehen.
Ausschalten	Erzwingen Sie ein Herunterfahren des virtuellen Rechners, und behalten Sie die Inhalte seiner Disk bei. Dies entspricht der Stromabschaltung bei einem physischen Rechner.
Zurücksetzen	Erzwingen Sie das Herunterfahren des virtuellen Rechners, und setzen Sie die Disk in den Ursprungszustand zurück. Achtung: Sämtliche auf der Disk gespeicherte Arbeit geht verloren.

Wenn Sie die Kursumgebung auf ihren ursprünglichen Zustand beim Start des Kurses zurücksetzen möchten, können Sie auf DELETE klicken, um die gesamte Kursumgebung zu entfernen. Nach dem Löschen des Labs klicken Sie auf [CREATE], um einen neuen Satz von Kursraumsystemen bereitzustellen.



Warnung

Der Vorgang [DELETE] kann nicht rückgängig gemacht werden. Die von Ihnen bis zu diesem Zeitpunkt in der Kursumgebung vorgenommene Arbeit geht verloren.

Der Autostop-Timer

Die Registrierung bei Red Hat Online Learning ermöglicht Kursteilnehmern eine bestimmte Menge Zeit am Computer. Für den sparsamen Umgang mit der vorgegebenen Computerzeit verfügt der ROL-Kursraum über einen verknüpften Zählvorgang, der die Kursumgebung herunterfährt, wenn der Timer abgelaufen ist.

Um den Timer anzupassen, klicken Sie auf [+]. Damit fügen Sie eine Stunde zum Timer hinzu.
Beachten Sie, dass die maximale Zeit zwölf Stunden beträgt.

Landessprachliche Anpassung

Sprachauswahl auf Benutzerbasis

Ihre Benutzer bevorzugen für ihre Desktop-Umgebung eventuell eine andere Sprache als die Standardsprache des Systems. Für ihr Benutzerkonto möchten sie möglicherweise auch ein anderes Tastaturlayout oder eine andere Eingabemethode verwenden.

Spracheinstellungen

In der GNOME-Desktop-Umgebung wird der Benutzer möglicherweise bei der ersten Anmeldung aufgefordert, seine bevorzugte Sprache und Eingabemethode einzustellen. Ist dies nicht der Fall, ist die Applikation Region & Language für den einzelnen Benutzer der einfachste Weg, die bevorzugte Sprache und Eingabemethode anzupassen.

Sie können diese Applikation auf zwei Arten starten. Sie können den Befehl `gnome-control-center region` über ein Terminal ausführen. Wählen Sie alternativ in der oberen Leiste in der rechten Ecke im Systemmenü die Schaltfläche für die Einstellungen (das Symbol mit dem gekreuzten Schraubendreher und Schraubenschlüssel) links unten im Menü aus.

Wählen Sie in dem sich öffnenden Fenster Region & Language aus. Klicken Sie auf das Feld Language, und wählen Sie aus der daraufhin angezeigten Liste die bevorzugte Sprache aus. Dadurch wird auch die Einstellung für Formats auf die Standardwerte dieser Sprache aktualisiert. Bei Ihrer nächsten Anmeldung werden die Änderungen vollständig wirksam.

Diese Einstellungen wirken sich auf die GNOME-Desktop-Umgebung sowie auf alle Applikationen wie beispielsweise `gnome-terminal` aus, die in der Umgebung gestartet werden. Standardmäßig werden sie jedoch nicht auf dieses Benutzerkonto angewendet, wenn über eine ssh-Anmeldung von einem Remote-System oder über eine textbasierte Anmeldung von einer virtuellen Konsole (beispielsweise `tty5`) darauf zugegriffen wird.



Anmerkung

Sie können festlegen, dass Ihre Shell-Umgebung dieselbe LANG-Einstellung wie Ihre grafische Umgebung verwendet, selbst wenn Sie sich über eine textbasierte virtuelle Konsole oder über ssh anmelden. Hierzu kann beispielsweise ein Code ähnlich dem folgenden in Ihrer `~/.bashrc`-Datei eingefügt werden. Der Code im folgenden Beispiel stellt die Sprache für eine Textanmeldung so ein, dass sie mit der zurzeit für die GNOME-Desktop-Umgebung des Benutzers konfigurierten Sprache übereinstimmt:

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

Japanisch, Koreanisch, Chinesisch und andere Sprachen mit nicht lateinischem Zeichensatz werden in textbasierten virtuellen Konsolen eventuell nicht einwandfrei angezeigt.

Einführung

Mithilfe einzelner Befehle kann die Verwendung einer anderen Sprache festgelegt werden, indem die Variable `LANG` in die Befehlszeile gesetzt wird:

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

Bei den nachfolgenden Befehlen erfolgt die Ausgabe wieder in der Standardsprache des Systems. Mit dem Befehl `Locale` können der aktuelle Wert von `LANG` und andere relevante Umgebungsvariablen festgelegt werden.

Einstellungen der Eingabemethode

GNOME 3 in Red Hat Enterprise Linux 7 oder höher verwendet automatisch das Auswahlsystem der Eingabemethode `IBus`, wodurch sich die Tastaturlayouts und Eingabemethoden schnell wechseln lassen.

Die Anwendung `Region & Language` kann auch zum Aktivieren alternativer Eingabemethoden verwendet werden. Im Applikationsfenster von `Region & Language` zeigt das Feld [`Input Sources`] an, welche Eingabemethoden derzeit verfügbar sind. Standardmäßig ist eventuell [`English (US)`] die einzige verfügbare Methode. Markieren Sie [`English (US)`], und klicken Sie auf das [`Tastatur`]-Symbol, um das aktuelle Tastaturlayout anzuzeigen.

Um eine weitere Eingabemethode hinzuzufügen, klicken Sie auf die Schaltfläche `[+]` im linken unteren Bereich des Fensters [`Input Sources`]. Das Fenster [`Add an Input Source`] wird geöffnet. Wählen Sie Ihre Sprache und anschließend die bevorzugte Eingabemethode oder das Tastaturlayout aus.

Wenn mehr als eine Eingabemethode konfiguriert ist, kann der Benutzer rasch zwischen diesen wechseln, indem er `[Super+Space]` (manchmal auch `[Windows+Space]`) eingibt. Auf der oberen GNOME-Leiste wird ein Statusindikator angezeigt, der über zwei Funktionen verfügt: Er gibt an, welche Ausgabemethode aktiv ist, und fungiert als Menü, das zum Wechseln zwischen Eingabemethoden oder zur Auswahl von erweiterten komplexeren Eingabemethoden verwendet werden kann.

Einige der Methoden sind mit Zahnrädern gekennzeichnet. Diese verfügen über erweiterte Konfigurationsoptionen und Funktionen. Beispielsweise kann der Benutzer mit der japanischen Eingabemethode [`Japanese (Kana Kanji)`] Text im lateinischen Zeichensatz vorab bearbeiten und mit den Tasten [`Pfeil nach unten`] und [`Pfeil nach oben`] die zu verwendenden Zeichen auswählen.

Englischsprachige Benutzer in den USA finden dies ggf. ebenfalls hilfreich. Bei der Eingabemethode [`English (United States)`] lautet beispielsweise das Tastaturlayout [`English (international AltGr dead keys)`], das die Taste `[AltGr]` (oder die rechte `[Alt]`-Taste) auf einer PC-Tastatur mit 104/105 Tasten als Zusatztaste in Form einer „zweiten Umschalttaste“ sowie als Aktivierungstaste für nicht belegte Tasten zur Eingabe zusätzlicher Zeichen behandelt. Zur Verfügung stehen auch Dvorak und andere Tastaturlayouts.



Anmerkung

In der GNOME Desktop-Umgebung können alle Unicode-Zeichen eingegeben werden, sofern Sie den Unicode-Codepunkt oder Unicode-Zahlenwert des Zeichens kennen. Drücken Sie [Strg+Umschalt+U], und geben Sie dann den Codepoint ein. Nach dem Drücken von [Strg+Umschalt+U] erscheint ein unterstrichenes u, das angibt, dass das System auf die Eingabe des Unicode-Codepunkts wartet.

Beispielsweise hat der kleine griechische Buchstabe Lambda den Codepoint U +03BB und wird durch Eingabe von [Strg+Umschalt+U], anschließend 03BB und Drücken der [Eingabetaste] eingegeben.

Einstellungen der Standardsprache des Systems

Die Standardsprache des Systems ist US-Englisch mit den Unicode-Zeichen der UTF-8-Codierung (`en_US.utf8`). Dies lässt sich aber während oder nach der Installation ändern.

Der Benutzer `root` kann in der Befehlszeile die systemweiten Ländereinstellungen mit dem Befehl `localectl` ändern. Wenn `localectl` ohne Argumente ausgeführt wird, werden die aktuellen Ländereinstellungen des Systems angezeigt.

Führen Sie zum Einstellen der systemweiten Standardsprache den Befehl `localectl set-locale LANG=locale` aus, wobei `locale` der entsprechende Wert für die LANG-Umgebungsvariable aus der Tabelle „Sprachcodereferenz“ in diesem Kapitel ist. Die Änderung wird für die Benutzer bei der nächsten Anmeldung wirksam und in der Datei `/etc/locale.conf` gespeichert.

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

In GNOME kann ein Administrator diese Einstellung ändern, indem er in `Region & Language` auf die Schaltfläche `[Login Screen]` in der rechten oberen Ecke des Fensters klickt. Das Ändern der `[Language]` des grafischen Anmeldebildschirms wirkt sich auch auf die Einstellungen der Standardsprache des Systems aus, die in der Konfigurationsdatei `/etc/locale.conf` gespeichert sind.



Wichtig

Textbasierte virtuelle Konsolen wie `tty4` sind hinsichtlich der anzeigbaren Schriftarten eingeschränkter als Terminals in einer virtuellen Konsole, die in einer grafischen Umgebung ausgeführt wird, oder als Pseudoterminals für ssh-Sitzungen. Japanische, koreanische und chinesische Zeichen beispielsweise werden in einer textbasierten virtuellen Konsole eventuell nicht richtig angezeigt. Daher sollten Sie in Betracht ziehen, Englisch oder eine andere Sprache mit einem lateinischen Zeichensatz als systemweite Standardeinstellung zu verwenden.

Gleichermaßen sind textbasierte virtuelle Konsolen bei den von ihnen unterstützten Eingabemethoden eingeschränkt. Dies wird separat über die grafische Desktop-Umgebung verwaltet. Die verfügbaren globalen Eingabeeinstellungen werden sowohl für textbasierte virtuelle Konsolen als auch für die grafische Umgebung anhand von `localectl` konfiguriert. Weitere Informationen finden Sie auf den Manpages `localectl(1)` und `vconsole.conf(5)`.

Sprachpakete

Mit speziellen RPM-Paketen, die als langpacks bezeichnet werden, werden Sprachpakete installiert, die Unterstützung für bestimmte Sprachen hinzufügen. Diese Sprachpakete nutzen Abhängigkeiten, um zusätzliche RPM-Pakete, die Lokalisierungen, Verzeichnisse und Übersetzungen für andere Softwarepakete enthalten, automatisch auf Ihrem System zu installieren.

Verwenden Sie `yum list langpacks-*` zum Auflisten der Sprachpakete, die installiert sind und möglicherweise installiert werden können:

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

Um Sprachunterstützung hinzuzufügen, installieren Sie das entsprechende Sprachpaket. Mit dem folgenden Befehl wird beispielsweise Unterstützung für Französisch hinzugefügt:

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

Mit `yum repoquery --whatsonplements` bestimmen Sie, welche RPM-Pakete durch ein Sprachpaket installiert werden können:

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



Wichtig

Sprachpakete nutzen schwache Abhängigkeiten für RPM, damit Zusatzpakete nur installiert werden, wenn das Kernpaket, das sie benötigt, ebenfalls installiert ist.

Wenn beispielsweise *langpacks-fr* wie in den vorherigen Beispielen gezeigt installiert wird, wird das Paket *mythes-fr* nur installiert, wenn auch der Thesaurus *mythes* auf dem System installiert ist.

Wenn *mythes* anschließend auf dem System installiert wird, wird das Paket *mythes-fr* aufgrund der schwachen Abhängigkeit des bereits installierten Pakets *langpacks-fr* ebenfalls automatisch installiert.



Literaturhinweise

Manpages `locale(7)`, `localectl(1)`, `locale.conf(5)`, `vconsole.conf(5)`, `unicode(7)` und `utf-8(7)`

Konvertierungen zwischen den Namen der X11-Layouts der grafischen Desktop-Umgebung und deren Namen in `localectl` befinden sich in der Datei `/usr/share/X11/xkb/rules/base.lst`.

Sprachcodereferenz



Anmerkung

Diese Tabelle enthält möglicherweise nicht alle auf Ihrem System verfügbaren Sprachpakete. Verwenden Sie `yum info langpacks-SUFFIX`, um weitere Informationen zu bestimmten Sprachpaketen abzurufen.

Sprachcodes

Sprache	Suffix für Sprachpakete	\$LANG-Wert
Englisch (US)	en	en_US.utf8
Assamesisch	as	as_IN.utf8
Bengalisch	bn	bn_IN.utf8
Chinesisch (vereinfacht)	zh_CN	zh_CN.utf8
Chinesisch (traditionell)	zh_TW	zh_TW.utf8
Französisch	fr	fr_FR.utf8
Deutsch	de	de_DE.utf8
Gujarati	gu	gu_IN.utf8
Hindi	hi	hi_IN.utf8
Italienisch	it	it_IT.utf8
Japanisch	ja	ja_JP.utf8
Kanareisch	kn	kn_IN.utf8
Koreanisch	ko	ko_KR.utf8

Sprache	Suffix für Sprachpakete	\$LANG-Wert
Malayalam	ml	ml_IN.utf8
Marathisch	mr	mr_IN.utf8
Odia	or	or_IN.utf8
Portugiesisch (Brasilien)	pt_BR	pt_BR.utf8
Punjabi	pa	pa_IN.utf8
Russisch	ru	ru_RU.utf8
Spanisch	es	es_ES.utf8
Tamilisch	ta	ta_IN.utf8
Telugu	te	te_IN.utf8

Kapitel 1

Introducing Container Technology

Ziel

Beschreiben, wie Anwendungen in von Red Hat OpenShift Container Platform orchestrierten Containern ausgeführt werden können.

Ziele

- Beschreiben des Unterschieds zwischen Container-Anwendungen und herkömmlichen Bereitstellungen
- Beschreiben der Container-Architekturgrundlagen
- Beschreiben der Vorteile von Orchestrierungsanwendungen und von OpenShift Container Platform

Abschnitte

- Überblick über die Container-Technologie (und Test)
- Überblick über die Container-Architektur (und Test)
- Überblick über Kubernetes und OpenShift (und Test)

Überblick über die Container-Technologie

Ziele

Nach Abschluss dieses Abschnitts sind die Teilnehmer in der Lage, den Unterschied zwischen Container-Anwendungen und herkömmlichen Bereitstellungen zu beschreiben.

Containerisierte Anwendungen

Softwareanwendungen hängen in der Regel von anderen Bibliotheken, Konfigurationsdateien oder Diensten ab, die von der Laufzeitumgebung bereitgestellt werden. Die herkömmliche Laufzeitumgebung für eine Softwareanwendung ist ein physischer Host oder ein virtueller Rechner, und Anwendungsabhängigkeiten werden als Teil des Hosts installiert.

Stellen Sie sich beispielsweise eine Python-Anwendung vor, die Zugriff auf eine allgemeine gemeinsam genutzte Bibliothek erfordert, die das TLS-Protokoll implementiert. Normalerweise installiert ein Systemadministrator das erforderliche Paket, das die gemeinsam genutzte Bibliothek bereitstellt, bevor die Python-Anwendung installiert wird.

Der größte Nachteil einer herkömmlich implementierten Softwareanwendung besteht darin, dass die Abhängigkeiten der Anwendung mit der Laufzeitumgebung verflochten sind.

Eine Anwendung kann beschädigt werden, wenn Updates oder Patches auf das Basisbetriebssystem (OS) angewendet werden.

Beispielsweise entfernt ein Betriebssystemupdate für die gemeinsam genutzte TLS-Bibliothek TLS 1.0 als unterstütztes Protokoll. Dadurch wird die implementierte Python-Anwendung beschädigt, da sie geschrieben wurde, um das TLS 1.0-Protokoll für Netzwerkanforderungen zu verwenden. Daraufhin wird der Systemadministrator gezwungen, ein Rollback des Betriebssystemupdates durchzuführen, damit die Anwendung weiterhin ausgeführt wird, sodass andere Anwendungen die Vorteile des aktualisierten Pakets nicht nutzen können.

Aus diesem Grund müssen Unternehmen, die herkömmliche Softwareanwendungen entwickeln, unter Umständen eine Reihe von Tests durchführen, um sicherzustellen, dass sich Aktualisierungen des Betriebssystems nicht negativ auf die auf dem Host ausgeführten Anwendungen auswirken.

Darüber hinaus muss eine herkömmlich bereitgestellte Anwendung gestoppt werden, bevor die zugehörigen Abhängigkeiten aktualisiert werden. Zum Minimieren von Anwendungsausfallzeiten entwerfen und implementieren Organisationen komplexe Systeme, um eine hohe Verfügbarkeit ihrer Anwendungen zu gewährleisten. Das Verwalten mehrerer Anwendungen auf einem einzelnen Host ist häufig umständlich. Zudem besteht bei jeder Bereitstellung oder Aktualisierung die Möglichkeit, dass eine der Anwendungen der Organisation beschädigt wird.

Abbildung 1.1 beschreibt den Unterschied zwischen Anwendungen, die als Container ausgeführt werden, und Anwendungen, die auf dem Host-Betriebssystem ausgeführt werden.

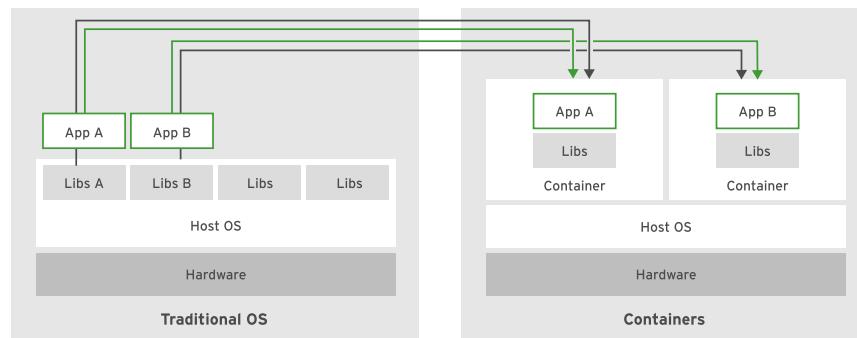


Abbildung 1.1: Unterschiede zwischen Container und Betriebssystem

Alternativ kann eine Softwareanwendung mithilfe eines Containers bereitgestellt werden.

Ein Container ist ein Satz aus einem oder mehreren Prozessen, die vom Rest des Systems isoliert sind.

Container bieten viele der Vorteile virtueller Rechner wie Sicherheit, Storage und Netzwerkalosolation. Sie benötigen aber viel weniger Hardweareressourcen und lassen sich schnell starten und beenden. Zudem isolieren sie die Bibliotheken und Laufzeitressourcen (wie CPU und Storage) für eine Anwendung zur Minimierung der Auswirkungen durch Aktualisierungen des Betriebssystems im Host-Betriebssystem, wie in Abbildung 1.1 beschrieben.

Die Verwendung von Containern ist nicht nur in Bezug auf Effizienz, Elastizität und Wiederverwendung der gehosteten Anwendungen hilfreich, sondern auch bei der Anwendungsportabilität. Die Open Container Initiative (OCI) stellt eine Reihe von Branchenstandards bereit, die eine Spezifikation für Container-Laufzeit und Container-Image definieren. Die Image-Spezifikation definiert das Format für das Bundle von Dateien und Metadaten, die ein Container-Image bilden. Wenn Sie eine Anwendung als Container-Image erstellen, die dem OCI-Standard entspricht, können Sie zum Ausführen der Anwendung eine beliebige OCI-konforme Container-Engine verwenden.

Es sind viele Container-Engines zum Verwalten und Ausführen einzelner Container verfügbar, einschließlich Rocket, Drawbridge, LXC, Docker und Podman. Podman ist in Red Hat Enterprise Linux 7.6 und höher verfügbar und wird in diesem Kurs zum Starten, Verwalten und Beenden einzelner Container verwendet.

Nachfolgend sind weitere wichtige Vorteile bei der Verwendung von Containern aufgeführt:

Niedriger Hardware-Footprint

Container verwenden interne Features des Betriebssystems zur Erstellung einer isolierten Umgebung, in der Ressourcen mithilfe von Betriebssystemfunktionen wie Namespaces und cGroups verwaltet werden. Bei diesem Ansatz wird im Gegensatz zu einem Hypervisor für virtuelle Rechner der CPU- und Speicher-Overhead minimiert. Durch die Ausführung einer Anwendung in einem virtuellen Rechner kann eine Isolation von der ausgeführten Umgebung erzielt werden. Dies erfordert jedoch eine Vielzahl an Services, die ebenso wie die Container diese Isolation unterstützen.

Isolation der Umgebung

Container funktionieren in einer geschlossenen Umgebung, in der Änderungen am Host-Betriebssystem oder anderen Anwendungen sich nicht auf den Container auswirken.

Da die vom Container benötigten Libraries eigenständig sind, kann die Anwendung ohne Unterbrechungen ausgeführt werden. Beispiel: Jede Anwendung kann in ihrem eigenen Container mit einer eigenen Library-Gruppe vorhanden sein. An einem Container vorgenommene Aktualisierungen wirken sich nicht auf andere Container aus.

Schnelle Bereitstellung

Container werden schnell bereitgestellt, da nicht das gesamte zugrunde liegende Betriebssystem installiert werden muss. In der Regel muss bei einer Isolation das Betriebssystem auf einem physischen Host oder virtuellen Rechner neu installiert werden. Auch eine einfache Aktualisierung kann einen vollständigen Neustart des Betriebssystems erfordern. Für einen Container-Neustart ist es nicht erforderlich, Services auf dem Host-Betriebssystem anzuhalten.

Bereitstellung mit mehreren Umgebungen

Bei einer herkömmlichen Bereitstellung mit einem einzelnen Host können Unterschiede zwischen Umgebungen die Anwendung kompromittieren. Bei der Verwendung von Containern werden jedoch alle Anwendungsabhängigkeiten und Umgebungseinstellungen im Container-Image gekapselt.

Wiederverwendbarkeit

Derselbe Container kann erneut verwendet werden, ohne dass eine vollständige Einrichtung des Betriebssystems erforderlich wird. Beispielsweise kann derselbe Datenbank-Container, der einen Produktionsdatenbankservice bereitstellt, während der Anwendungsentwicklung von jedem Entwickler zum Erstellen einer Entwicklungsdatenbank verwendet werden.

Wenn Container verwendet werden, müssen keine separaten Produktions- und Entwicklungsdatenbankserver verwaltet werden. Ein einzelnes Container-Image wird zum Erstellen von Instanzen des Datenbankservices verwendet.

Häufig werden Softwareanwendungen mit allen zugehörigen Services (Datenbanken, Messaging, Dateisysteme) in einem einzelnen Container ausgeführt. Dies kann zu den gleichen Problemen führen, die bei herkömmlichen Softwarebereitstellungen auf virtuellen Rechnern oder physischen Hosts auftreten. In solchen Fällen ist eine Bereitstellung mit mehreren Containern möglicherweise geeigneter.

Container sind außerdem ein idealer Ansatz, wenn Microservices für die Anwendungsentwicklung verwendet werden. Jeder Service ist in einer einfachen und zuverlässigen Container-Umgebung gekapselt, die in einer Produktions- oder Entwicklungsumgebung bereitgestellt werden kann. Die von einer Anwendung benötigte Sammlung von Container-Services kann auf einem einzigen Computer gehostet werden, sodass nicht für jeden Service ein Computer verwaltet werden muss.

Im Gegensatz dazu sind viele Anwendungen für eine Container-Umgebung nicht gut geeignet. Beispielsweise sind Anwendungen, die auf hardwarenahe Informationen zugreifen, beispielsweise Arbeitsspeicher, Dateisysteme und Geräte, aufgrund von Container-Einschränkungen möglicherweise unzuverlässig.



Literaturhinweise

Start - Open Containers Initiative

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

► Quiz

Überblick über die Container-Technologie

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

- 1. Welche beiden Optionen sind Beispiele für Softwareanwendungen, die in einem Container ausgeführt werden können? (Wählen Sie zwei Antworten aus.)
- a. Eine datenbankgesteuerte Python-Anwendung, die über einen einzelnen physischen Host auf Services zugreift, beispielsweise MySQL-Datenbanken, File Transfer Protocol-Server (FTP) und Webserver
 - b. Eine Java Enterprise Edition-Anwendung mit einer Oracle-Datenbank und einem Message Broker, die auf einem einzelnen virtuellen Rechner ausgeführt werden
 - c. Ein E/A-Überwachungstool, das für die Analyse des Datenverkehrs und die Blockierung der Datenübertragung zuständig ist
 - d. Ein Anwendungstool für Speicher-Images, das Snapshots aus allen CPU-Speichercaches für Debugging-Zwecke erstellen kann
- 2. Welche zwei der folgenden Anwendungsfälle sind für Container am besten geeignet? (Wählen Sie zwei Antworten aus.)
- a. Ein Softwareanbieter muss Software verteilen, die von anderen Unternehmen schnell und fehlerfrei wiederverwendet werden kann.
 - b. Ein Unternehmen stellt Anwendungen auf einem physischen Host bereit und möchte die Leistung mithilfe von Containern verbessern.
 - c. Entwickler in einem Unternehmen benötigen eine austauschbare Umgebung, die die Produktionsumgebung imitiert, sodass sie schnell den Code testen können, den sie entwickeln.
 - d. Ein Finanzunternehmen implementiert ein rechenintensives Risikoanalysetool auf seinen eigenen Containern, um die Anzahl der benötigten Prozessoren zu minimieren.

► 3. Ein Unternehmen migriert seine PHP-und Python-Anwendungen, die auf demselben Host ausgeführt werden, zu einer neuen Architektur. Aufgrund von internen Richtlinien verwenden beide eine Reihe von benutzerdefinierten, gemeinsam genutzten Bibliotheken aus dem Betriebssystem. Die letzte Aktualisierung, die nach einer Anforderung durch das Entwicklungsteam von Python auf die Bibliotheken angewendet wurde, hat zu einer Beschädigung der PHP-Anwendung geführt. Welche beiden Architekturen bieten die beste Unterstützung für beide Anwendungen?

(Wählen Sie zwei Antworten aus.)

- a. Bereitstellung der Anwendungen auf unterschiedlichen virtuellen Rechnern und individuelle Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf den Host des jeweiligen virtuellen Rechners
- b. Bereitstellung der Anwendungen auf unterschiedlichen Containern und individuelle Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf dem jeweiligen Container
- c. Bereitstellung der Anwendungen auf unterschiedlichen virtuellen Rechnern und Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf alle Hosts des virtuellen Rechners
- d. Bereitstellung der Anwendungen auf unterschiedlichen Containern und Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf alle Container

► 4. Welche drei Anwendungen können zur sofortigen Nutzung als Container verpackt werden? (Wählen Sie drei Antworten aus.)

- a. Hypervisor für virtuelle Rechner
- b. Blog-Software, beispielsweise WordPress
- c. Eine Datenbank
- d. Lokales Tool für die Dateisystem-Wiederherstellung
- e. Webserver

► Lösung

Überblick über die Container-Technologie

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

- 1. Welche beiden Optionen sind Beispiele für Softwareanwendungen, die in einem Container ausgeführt werden können? (Wählen Sie zwei Antworten aus.)
- a. Eine datenbankgesteuerte Python-Anwendung, die über einen einzelnen physischen Host auf Services zugreift, beispielsweise MySQL-Datenbanken, File Transfer Protocol-Server (FTP) und Webserver
 - b. Eine Java Enterprise Edition-Anwendung mit einer Oracle-Datenbank und einem Message Broker, die auf einem einzelnen virtuellen Rechner ausgeführt werden
 - c. Ein E/A-Überwachungstool, das für die Analyse des Datenverkehrs und die Blockierung der Datenübertragung zuständig ist
 - d. Ein Anwendungstool für Speicher/Images, das Snapshots aus allen CPU-Speichercaches für Debugging-Zwecke erstellen kann
- 2. Welche zwei der folgenden Anwendungsfälle sind für Container am besten geeignet? (Wählen Sie zwei Antworten aus.)
- a. Ein Softwareanbieter muss Software verteilen, die von anderen Unternehmen schnell und fehlerfrei wiederverwendet werden kann.
 - b. Ein Unternehmen stellt Anwendungen auf einem physischen Host bereit und möchte die Leistung mithilfe von Containern verbessern.
 - c. Entwickler in einem Unternehmen benötigen eine austauschbare Umgebung, die die Produktionsumgebung imitiert, sodass sie schnell den Code testen können, den sie entwickeln.
 - d. Ein Finanzunternehmen implementiert ein rechenintensives Risikoanalysetool auf seinen eigenen Containern, um die Anzahl der benötigten Prozessoren zu minimieren.

► 3. Ein Unternehmen migriert seine PHP-und Python-Anwendungen, die auf demselben Host ausgeführt werden, zu einer neuen Architektur. Aufgrund von internen Richtlinien verwenden beide eine Reihe von benutzerdefinierten, gemeinsam genutzten Bibliotheken aus dem Betriebssystem. Die letzte Aktualisierung, die nach einer Anforderung durch das Entwicklungsteam von Python auf die Bibliotheken angewendet wurde, hat zu einer Beschädigung der PHP-Anwendung geführt. Welche beiden Architekturen bieten die beste Unterstützung für beide Anwendungen?

(Wählen Sie zwei Antworten aus.)

- a. Bereitstellung der Anwendungen auf unterschiedlichen virtuellen Rechnern und individuelle Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf den Host des jeweiligen virtuellen Rechners
- b. Bereitstellung der Anwendungen auf unterschiedlichen Containern und individuelle Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf dem jeweiligen Container
- c. Bereitstellung der Anwendungen auf unterschiedlichen virtuellen Rechnern und Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf alle Hosts des virtuellen Rechners
- d. Bereitstellung der Anwendungen auf unterschiedlichen Containern und Anwendung der benutzerspezifischen, gemeinsam genutzten Bibliotheken auf alle Container

► 4. Welche drei Anwendungen können zur sofortigen Nutzung als Container verpackt werden? (Wählen Sie drei Antworten aus.)

- a. Hypervisor für virtuelle Rechner
- b. Blog-Software, beispielsweise WordPress
- c. Eine Datenbank
- d. Lokales Tool für die Dateisystem-Wiederherstellung
- e. Webserver

Überblick über die Container-Architektur

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Beschreibung der Architektur von Linux-Containern
- Beschreibung des Podman-Tools für die Verwaltung von Containern.

Einführung in die Container-Geschichte

Container haben in den letzten Jahren schnell an Beliebtheit gewonnen. Die Technologie, die Containern zugrunde liegt, gibt es jedoch schon relativ lange. Im Jahr 2001 führte Linux das Projekt „VServer“ ein. VServer war der erste Versuch, vollständige Prozesssätze innerhalb eines einzelnen Servers mit einem hohen Isolationsgrad auszuführen.

Ausgehend von VServer wurde die Idee der isolierten Prozesse weiterentwickelt und um die folgenden Funktionen des Linux-Kernels herum formalisiert:

Namespaces

Ein Namespace isoliert spezifische Systemressourcen, die normalerweise für alle Prozesse sichtbar sind. In einem Namespace können nur Prozesse, die Teil dieses Namespaces sind, diese Ressourcen anzeigen. In Namespaces können Ressourcen enthalten sein. Dazu zählen beispielsweise Netzwerkschnittstellen, die Liste der Prozess-IDs, Mount-Punkte, IPC-Ressourcen und die Informationen zum Hostnamen des Systems.

Kontrollgruppen (cGroups)

Kontrollgruppen partitionieren Prozesse und die zugehörigen untergeordneten Prozesse in Gruppen, um die verbrauchten Ressourcen zu verwalten und einzuschränken. Kontrollgruppen schränken die Anzahl der Systemressourcen ein, die von Prozessen verwendet werden können. Diese Einschränkungen verhindern, dass ein Prozess zu viele Ressourcen auf dem Host verwendet.

Seccomp

Das 2005 entwickelte und etwa 2014 in Containern eingeführte Seccomp beschränkt, wie Prozesse Systemaufrufe verwenden können. Seccomp definiert ein Sicherheitsprofil für Prozesse und setzt die Systemaufrufe, Parameter und Dateideskriptoren auf die Positivliste, die von ihnen verwendet werden dürfen.

SELinux

Security-Enhanced Linux (SELinux) ist ein obligatorisches Zugangskontrollsystem für Prozesse. Der Linux-Kernel verwendet SELinux, um Prozesse voreinander zu schützen und um das Hostsystem vor den in Ausführung befindlichen Prozessen zu schützen. Prozesse werden als „eingeschränkter“ SELinux-Typ ausgeführt, der eingeschränkten Zugriff auf die Ressourcen des Hostsystems hat.

Alle diese Neuerungen und Features konzentrieren sich auf ein Grundkonzept: Prozesse können isoliert ausgeführt werden, während gleichzeitig auf Systemressourcen zugegriffen wird. Dieses Konzept ist die Grundlage der Container-Technologie und die Basis für alle Container-Implementierungen. Container sind heutzutage Prozesse im Linux-Kernel, die diese Sicherheitsfeatures verwenden, um eine isolierte Umgebung zu erstellen. Diese Umgebung verhindert, dass isolierte Prozesse System- oder andere Container-Ressourcen zweckentfremden.

Ein häufiger Anwendungsfall von Containern besteht darin, dass mehrere Replikate desselben Services (z. B. ein Datenbankserver) auf demselben Host vorhanden sind. Jedes Replikat verfügt über isolierte Ressourcen (Dateisystem, Ports, Arbeitsspeicher). Daher muss der Service keine Ressourcen gemeinsam nutzen. Durch die Isolierung wird garantiert, dass ein fehlerhafter oder schädlicher Service andere Services oder Container in demselben Host oder im darunter liegenden System nicht beeinträchtigt.

Beschreiben der Architektur von Linux-Containern

Aus der Sicht des Linux-Kernels ist ein Container ein Prozess mit Einschränkungen. Anstatt eine einzelne Binärdatei auszuführen, führt ein Container ein Image aus. Ein Image ist ein Dateisystempaket, das alle für die Ausführung eines Prozesses erforderlichen Abhängigkeiten enthält: Dateien im Dateisystem, installierte Pakete, verfügbare Ressourcen, laufende Prozesse und Kernelmodule.

Während ausführbare Dateien die Grundlage für die Ausführung von Prozessen bilden, bilden Images die Grundlage für die Ausführung von Containern. Laufende Container verwenden eine unveränderliche Ansicht des Images, sodass mehrere Container dasselbe Image gleichzeitig wiederverwenden können. Da es sich bei Images um Dateien handelt, können sie von Versionsverwaltungssystemen verwaltet werden, wodurch die Automatisierung der Bereitstellung von Containern und Images verbessert wird.

Container-Images müssen für die Container-Laufzeit lokal verfügbar sein, um sie ausführen zu können. Normalerweise werden die Images jedoch in einem Image-Repository gespeichert und verwaltet. Ein Image-Repository ist nur ein öffentlicher oder privater Service, in dem Images gespeichert, gesucht und abgerufen werden können. Zu den weiteren Features, die von Image-Repositories bereitgestellt werden, zählen der Remote-Zugriff, Image-Metadaten und die Autorisierung oder Versionskontrolle von Images.

Es gibt viele verschiedene Image-Repositories, die jeweils unterschiedliche Features bieten:

- 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/>]



Anmerkung

In diesem Kurs wird die öffentliche Image-Registry Quay verwendet. Daher können die Kursteilnehmer mit Images arbeiten, ohne sich Gedanken über gegenseitige Konflikte machen zu müssen.

Verwalten von Containern mit Podman

Container, Images und Image-Registries müssen miteinander agieren können. Beispielsweise müssen Sie in der Lage sein, Images zu erstellen und sie in Image-Registries abzulegen. Sie müssen auch in der Lage sein, ein Image aus der Image-Registry abzurufen und aus diesem Image einen Container zu erstellen.

Podman ist ein Open Source-Tool zum Verwalten von Containern und Container-Images und zum Interagieren mit Image-Registries. Es bietet die folgenden wichtigen Features:

Kapitel 1 | Introducing Container Technology

- Es verwendet das von der Open Container Initiative [<https://www.opencontainers.org>] (OCI) angegebene Image-Format. Diese Spezifikationen definieren ein standardmäßiges, von der Community gesteuertes proprietäres Image-Format.
- Podman speichert lokale Images im lokalen Dateisystem. Dadurch wird verhindert, dass eine Client-/Serverarchitektur erforderlich ist oder dass Daemons auf dem lokalen Rechner ausgeführt werden.
- Podman befolgt dieselben Befehlsmuster wie die Docker-CLI, sodass Sie sich nicht mit einem neuen Toolset vertraut machen müssen.
- Podman ist mit Kubernetes kompatibel. Kubernetes kann Podman zur Verwaltung seiner Container verwenden.



Literaturhinweise

Red Hat Quay Container Registry

<https://quay.io>

Podman-Site

<https://podman.io/>

Open Container Initiative

<https://www.opencontainers.org>

► Quiz

Überblick über die Container-Architektur

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

- ▶ 1. Welche drei der folgenden Linux-Funktionen werden zum Ausführen von Containern verwendet? (Wählen Sie drei Antworten aus.)
 - a. Namespaces
 - b. Integrity Management
 - c. Security-Enhanced Linux
 - d. Kontrollgruppen

- ▶ 2. Welche der folgenden Aussagen beschreibt ein Container-Image am treffendsten?
 - a. Image eines virtuellen Rechners, über das ein Container erstellt wird
 - b. Container-Blueprint, über das ein Container erstellt wird
 - c. Laufzeitumgebung, in der eine Anwendung ausgeführt wird
 - d. Indexdatei des Containers, die von einer Registry verwendet wird

- ▶ 3. Welche drei der folgenden Komponenten sind bei Implementierungen der Container-Architektur üblich? (Wählen Sie drei Antworten aus.)
 - a. Container-Laufzeit
 - b. Container-Berechtigungen
 - c. Container-Images
 - d. Container-Registries

- ▶ 4. Was ist ein Container in Bezug auf den Linux-Kernel?
 - a. Ein virtueller Rechner
 - b. Ein isolierter Prozess mit reguliertem Zugriff auf Ressourcen
 - c. Eine Reihe von Dateisystemebenen, die von UnionFS freigelegt werden
 - d. Ein externer Service, der Container-Images bereitstellt

► Lösung

Überblick über die Container-Architektur

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

- ▶ **1. Welche drei der folgenden Linux-Funktionen werden zum Ausführen von Containern verwendet? (Wählen Sie drei Antworten aus.)**
 - a. Namespaces
 - b. Integrity Management
 - c. Security-Enhanced Linux
 - d. Kontrollgruppen
- ▶ **2. Welche der folgenden Aussagen beschreibt ein Container-Image am treffendsten?**
 - a. Image eines virtuellen Rechners, über das ein Container erstellt wird
 - b. Container-Blueprint, über das ein Container erstellt wird
 - c. Laufzeitumgebung, in der eine Anwendung ausgeführt wird
 - d. Indexdatei des Containers, die von einer Registry verwendet wird
- ▶ **3. Welche drei der folgenden Komponenten sind bei Implementierungen der Container-Architektur üblich? (Wählen Sie drei Antworten aus.)**
 - a. Container-Laufzeit
 - b. Container-Berechtigungen
 - c. Container-Images
 - d. Container-Registries
- ▶ **4. Was ist ein Container in Bezug auf den Linux-Kernel?**
 - a. Ein virtueller Rechner
 - b. Ein isolierter Prozess mit reguliertem Zugriff auf Ressourcen
 - c. Eine Reihe von Dateisystemebenen, die von UnionFS freigelegt werden
 - d. Ein externer Service, der Container-Images bereitstellt

Überblick über Kubernetes und OpenShift

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Ermitteln der Einschränkungen von Linux-Containern und der Notwendigkeit der Container-Orchestrierung
- Beschreiben des Kubernetes-Container-Orchestrierungstools
- Beschreiben von Red Hat OpenShift Container Platform (RHOC)

Einschränkungen von Containern

Mit Containern können Services einfach verpackt und ausgeführt werden. Die steigende Anzahl der von einer wachsenden Organisation verwalteten Container führt zu einem exponentiellen Anstieg der zum manuellen Starten dieser Container erforderlichen Arbeit und der Notwendigkeit, schnell auf externe Anforderungen zu reagieren.

Bei der Verwendung von Containern in einer Produktionsumgebung benötigen Unternehmen häufig:

- Eine einfache Kommunikation zwischen einer Vielzahl von Services.
- Ressourcenobergrenzen für Anwendungen unabhängig von der Anzahl der Container, in denen sie ausgeführt werden.
- Reaktionen auf Anwendungsspitzen, um die Anzahl der laufenden Container zu erhöhen oder zu verringern.
- Reaktion auf die Verschlechterung des Services.
- Schrittweiser Rollout eines neuen Release für eine Reihe von Benutzern.

Unternehmen benötigen oft eine Container-Orchestrierungstechnologie, weil Container-Laufzeiten (wie Podman) die obigen Anforderungen nicht angemessen erfüllen.

Überblick über Kubernetes

Kubernetes ist ein Orchestrierungsservice, der die Bereitstellung, Verwaltung und Skalierung von Container-Anwendungen vereinfacht.

Die kleinste in Kubernetes verwaltbare Einheit ist ein Pod. Ein Pod besteht aus einem oder mehreren Containern mit Speicherressourcen und einer IP-Adresse, die eine einzelne Anwendung darstellen. Kubernetes verwendet Pods auch, um die Container darin zu orchestrieren und seine Ressourcen als eine einzelne Einheit zu beschränken.

Kubernetes-Features

Kubernetes bietet folgende Features, die auf einer Container-Infrastruktur basieren:

Service Discovery und Lastverteilung

Kubernetes ermöglicht eine serviceübergreifende Kommunikation durch das Zuweisen eines einzelnen DNS-Eintrags zu jedem Satz von Containern. Auf diese Weise muss der anfordernde Service nur den DNS-Namen des Ziels kennen, sodass der Cluster den Speicherort und die IP-Adresse des Containers ändern kann, ohne dass der Service davon betroffen ist. Dadurch kann die Last der Anforderung im gesamten Pool der Container verteilt werden, die den Service bereitstellen. Beispielsweise kann Kubernetes eingehende Anforderungen an einen MySQL-Service gleichmäßig aufteilen, wobei die Verfügbarkeit der Pods berücksichtigt wird.

Horizontale Skalierung

Mithilfe der Kubernetes-Befehlszeilenschnittstelle oder der Webbenutzeroberfläche können Anwendungen manuell oder automatisch mit einem Konfigurationssatz skaliert werden.

Selbstreparatur

Kubernetes kann zum Überwachen von Containern benutzerdefinierte Health Checks verwenden, um diese bei einem Ausfall neu zu starten und neu zu planen.

Automatisierter Rollout

Kubernetes kann Updates schrittweise auf die Container Ihrer Anwendungen verteilen und dabei ihren Status überprüfen. Wenn während des Rollouts Probleme auftreten, kann Kubernetes auf die vorherige Iteration der Bereitstellung zurückgreifen.

Geheimnis- und Konfigurationsverwaltung

Sie können die Konfigurationseinstellungen und Geheimnisse Ihrer Anwendungen verwalten, ohne Container neu erstellen zu müssen. Bei Anwendungsgeheimnissen kann es sich um Benutzernamen, Kennwörter und Serviceendpunkte handeln, also um Konfigurationseinstellungen, die privat gehalten werden müssen.

Operatoren

Operatoren sind Kubernetes-Anwendungspakete, die das Wissen über den Lebenszyklus der Anwendung in den Kubernetes-Cluster einbringen. Als Operatoren gepackte Anwendungen verwenden die Kubernetes-API, um den Status des Clusters zu aktualisieren und auf Änderungen des Anwendungsstatus zu reagieren.

Überblick über OpenShift

Red Hat OpenShift Container Platform (RHOP) umfasst modulare Komponenten und Services, die auf einer Kubernetes-Container-Infrastruktur basieren. RHOP fügt PaaS-Produktionsplattformfunktionen wie Remote-Management, Multi-Tenancy, erhöhte Sicherheit, Lifecycle-Management für Anwendungen und Self-Service-Schnittstellen für Entwickler hinzu.

Ab Red Hat OpenShift v4 verwenden Hosts in einem OpenShift-Cluster Red Hat Enterprise Linux CoreOS als das zugrunde liegende Betriebssystem.



Anmerkung

Im Verlauf dieses Kurses verweisen die Begriffe „RHOP“ und „OpenShift“ auf Red Hat OpenShift Container Platform.

OpenShift-Features

OpenShift fügt einem Kubernetes-Cluster die folgenden Features hinzu:

Integrierter Entwickler-Workflow

RHOCP integriert eine integrierte Container-Registry, CI/CD-Pipelines und S2I, ein Tool zum Erstellen von Artefakten aus Quell-Repositories in Container-Images.

Routen

Stellen der Außenwelt Services ohne Weiteres bereit.

Metriken und Protokollierung

Umfassen integrierte und Selbstanalyse-Metrikservices und aggregierte Protokollierung.

Einheitliche Benutzeroberfläche

OpenShift bietet vereinheitlichte Tools und eine Benutzeroberfläche zur Verwaltung der verschiedenen Funktionen.



Literaturhinweise

Produktionsreife Container-Orchestrierung – Kubernetes

<https://kubernetes.io/>

OpenShift: Container Application Platform by Red Hat, Built on Docker and Kubernetes

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

► Quiz

Beschreibung von Kubernetes und OpenShift

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

- 1. Welche drei der folgenden Aussagen sind in Bezug auf Container-Einschränkungen richtig? (Wählen Sie drei Antworten aus.)
- a. Container sind leicht in großer Anzahl zu orchestrieren.
 - b. Eine fehlende Automatisierung erhöht die Reaktionszeit auf Probleme.
 - c. Container verwalten nicht die in ihnen enthaltenen Anwendungsfehler.
 - d. Für Container liegt kein Load Balancing vor.
 - e. Container sind stark isolierte Anwendungspakete.
- 2. Welche zwei der folgenden Aussagen sind in Bezug auf Kubernetes richtig? (Wählen Sie zwei Antworten aus.)
- a. Kubernetes ist ein Container.
 - b. Kubernetes kann nur Docker-Container verwenden.
 - c. Kubernetes ist ein Container-Orchestrierungssystem.
 - d. Kubernetes vereinfacht die Verwaltung, Bereitstellung und Skalierung von Container-Anwendungen.
 - e. Anwendungen, die in einem Kubernetes-Cluster verwaltet werden, sind schwieriger zu warten.
- 3. Welche drei der folgenden Aussagen sind in Bezug auf Red Hat OpenShift v4 richtig? (Wählen Sie drei Antworten aus.)
- a. OpenShift bietet zusätzliche Features für eine Kubernetes-Infrastruktur.
 - b. Kubernetes und OpenShift schließen sich gegenseitig aus.
 - c. OpenShift-Hosts verwenden Red Hat Enterprise Linux als Basisbetriebssystem.
 - d. OpenShift vereinfacht die Entwicklung zum Implementieren von Source-to-Image-Technologien und CI/CD-Pipelines.
 - e. OpenShift vereinfacht das Routing und Load Balancing.
- 4. Welche Features bietet OpenShift, um die Kubernetes-Funktionen zu erweitern? (Wählen Sie zwei Antworten aus.)
- a. Operatoren und das Operator-Framework.
 - b. Routen zur externen Bereitstellung der Services.
 - c. Einen integrierten Entwicklungs-Workflow.
 - d. Selbstreparatur und Health Checks.

► Lösung

Beschreibung von Kubernetes und OpenShift

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

► 1. Welche drei der folgenden Aussagen sind in Bezug auf Container-Einschränkungen richtig? (Wählen Sie drei Antworten aus.)

- a. Container sind leicht in großer Anzahl zu orchestrieren.
- b. Eine fehlende Automatisierung erhöht die Reaktionszeit auf Probleme.
- c. Container verwalten nicht die in ihnen enthaltenen Anwendungsfehler.
- d. Für Container liegt kein Load Balancing vor.
- e. Container sind stark isolierte Anwendungspakete.

► 2. Welche zwei der folgenden Aussagen sind in Bezug auf Kubernetes richtig? (Wählen Sie zwei Antworten aus.)

- a. Kubernetes ist ein Container.
- b. Kubernetes kann nur Docker-Container verwenden.
- c. Kubernetes ist ein Container-Orchestrierungssystem.
- d. Kubernetes vereinfacht die Verwaltung, Bereitstellung und Skalierung von Container-Anwendungen.
- e. Anwendungen, die in einem Kubernetes-Cluster verwaltet werden, sind schwieriger zu warten.

► 3. Welche drei der folgenden Aussagen sind in Bezug auf Red Hat OpenShift v4 richtig? (Wählen Sie drei Antworten aus.)

- a. OpenShift bietet zusätzliche Features für eine Kubernetes-Infrastruktur.
- b. Kubernetes und OpenShift schließen sich gegenseitig aus.
- c. OpenShift-Hosts verwenden Red Hat Enterprise Linux als Basisbetriebssystem.
- d. OpenShift vereinfacht die Entwicklung zum Implementieren von Source-to-Image-Technologien und CI/CD-Pipelines.
- e. OpenShift vereinfacht das Routing und Load Balancing.

► 4. Welche Features bietet OpenShift, um die Kubernetes-Funktionen zu erweitern? (Wählen Sie zwei Antworten aus.)

- a. Operatoren und das Operator-Framework.
- b. Routen zur externen Bereitstellung der Services.
- c. Einen integrierten Entwicklungs-Workflow.
- d. Selbstreparatur und Health Checks.

► Angeleitete Übung

Konfigurieren der Kursumgebung

In dieser Übung konfigurieren Sie die Workstation für den Zugriff auf die gesamte Infrastruktur, die in diesem Kurs verwendet wird.

Ergebnisse

Es werden folgende Fähigkeiten vermittelt:

- Konfigurieren des workstation-Rechners für den Zugriff auf einen OpenShift-Cluster, eine Container-Image-Registry und ein Git-Repository, die im Kurs verwendet werden
- Verzweigen des Repository für die Beispielanwendungen des Kurses auf Ihr persönliches GitHub-Benutzerkonto
- Klonen dieses Repository für die Beispielanwendungen des Kurses von Ihrem persönlichen GitHub-Benutzerkonto auf Ihren workstation-Rechner

Bevor Sie Beginnen

Sie müssen über Folgendes verfügen, um diese Übung durchführen zu können:

- Zugriff auf den DO180-Kurs in der Online-Lernumgebung von Red Hat Training
- Die Verbindungsparameter und ein Entwicklerbenutzerkonto für den Zugriff auf einen OpenShift-Cluster, der von Red Hat Training verwaltet wird
- Ein persönliches, kostenloses GitHub-Benutzerkonto. Lesen Sie die Anweisungen in *Anhang B, Erstellen eines GitHub-Benutzerkontos*, wenn Sie sich bei GitHub registrieren müssen.
- Ein persönliches, kostenloses Quay.io-Benutzerkonto. Lesen Sie die Anweisungen in *Anhang C, Erstellen eines Quay-Benutzerkontos*, wenn Sie sich bei Quay.io registrieren müssen.
- Ein persönliches GitHub-Zugriffstoken.

Anweisungen

Vergewissern Sie sich vor dem Beginn einer Übung, dass Sie über Folgendes verfügen:

- 1. Bereiten Sie Ihr GitHub-Zugriffstoken vor.
- 1.1. Navigieren Sie mit einem Webbrowser zu <https://github.com>, und authentifizieren Sie sich.
 - 1.2. Klicken Sie oben auf der Seite auf Ihr Profilsymbol, wählen Sie das Menü **Settings** aus, und wählen Sie dann im linken Bereich der Seite **Developer settings** aus.

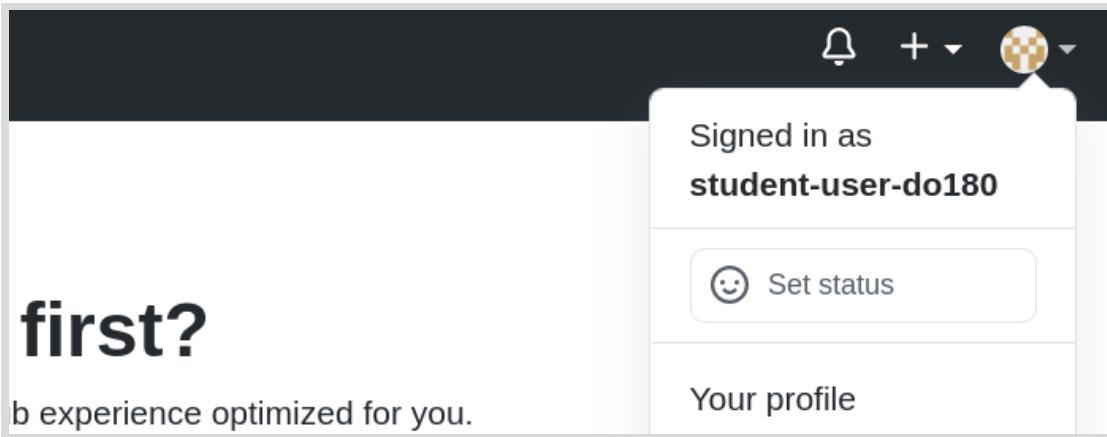


Abbildung 1.2: Benutzermenü

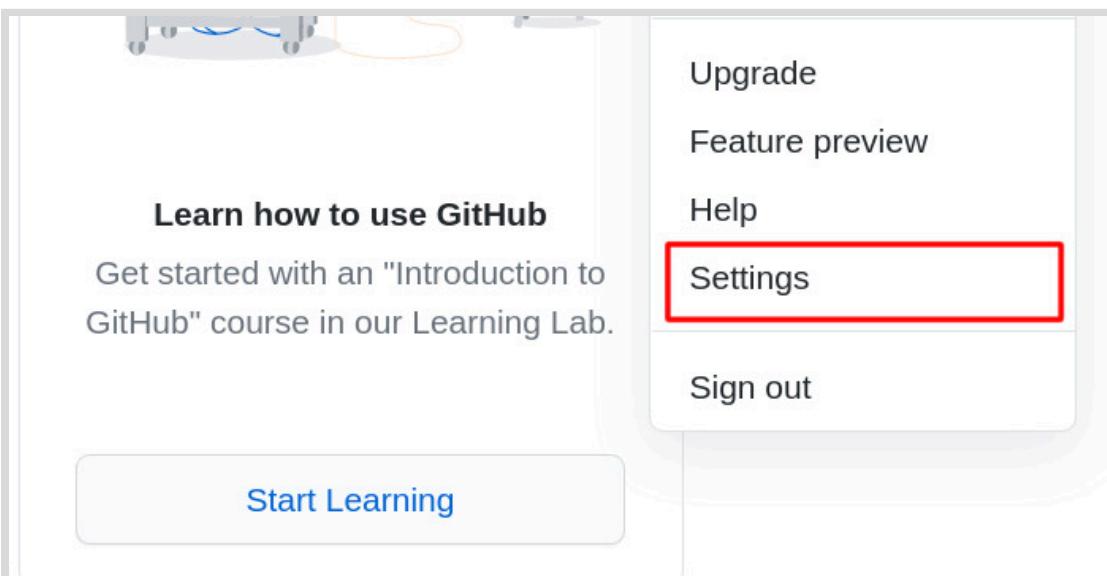


Abbildung 1.3: Menü „Settings“

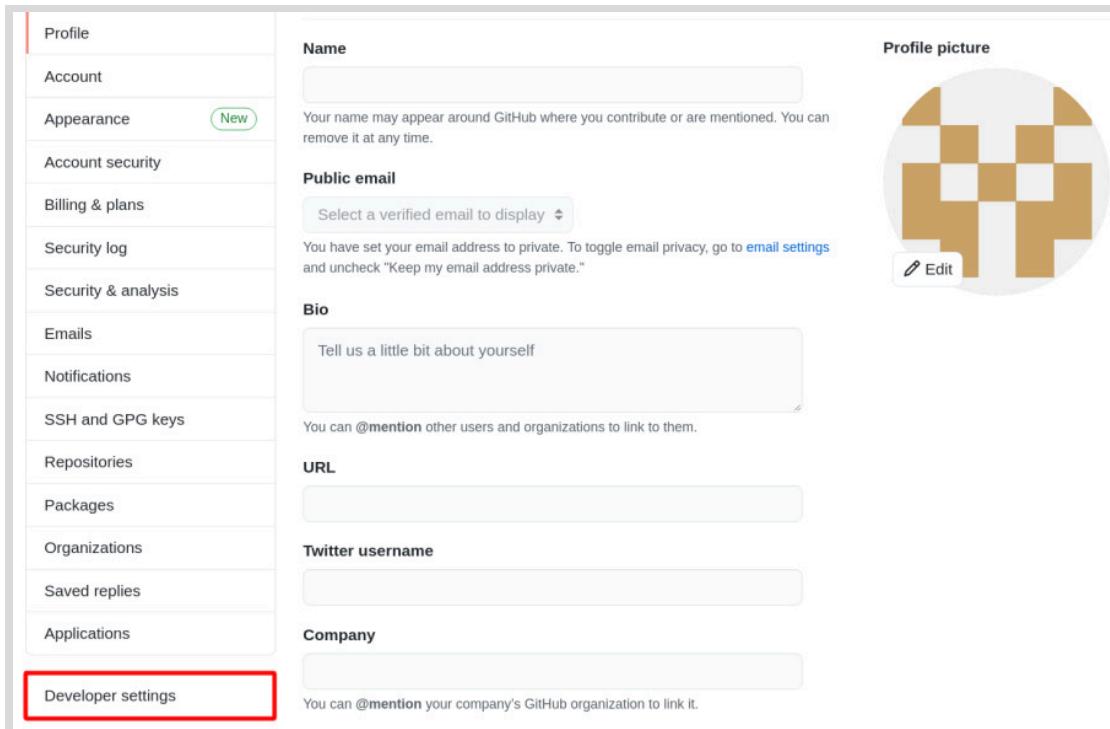


Abbildung 1.4: Developer settings

- 1.3. Wählen Sie im linken Bereich den Abschnitt mit dem persönlichen Zugriffstoken aus. Erstellen Sie auf der nächsten Seite Ihr neues Token, indem Sie auf **Generate new token** klicken. Sie werden dann aufgefordert, Ihr Passwort einzugeben.

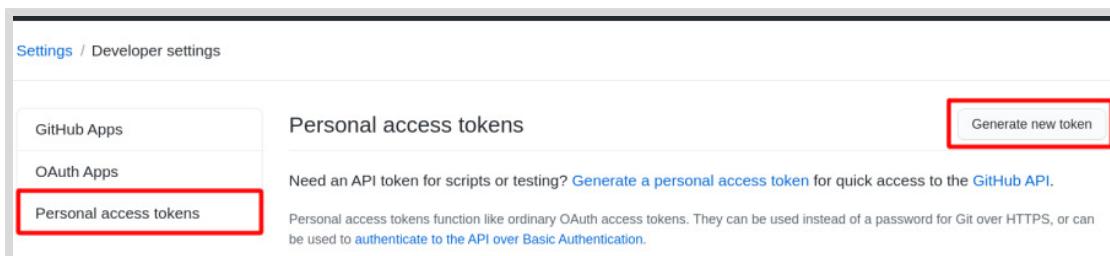


Abbildung 1.5: Bereich für persönliches Zugriffstoken

- 1.4. Geben Sie in das Feld **Note** eine kurze Beschreibung Ihres neuen Zugriffstokens ein.
- 1.5. Wählen Sie die Option **public_repo** aus, und lassen Sie die anderen Optionen deaktiviert. Erstellen Sie Ihr neues Zugriffstoken, indem Sie auf **Generate token** klicken.

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

Abbildung 1.6: Konfiguration des persönlichen Zugriffstokens

- 1.6. Ihr neues persönliches Zugriffstoken wird in der Ausgabe angezeigt. Erstellen Sie mit Ihrem bevorzugten Texteditor eine neue Datei im Benutzerverzeichnis des Kursteilnehmers mit dem Namen `token`, und fügen Sie Ihr generiertes persönliches Zugriffstoken ein. Das persönliche Zugriffstoken kann nicht erneut in GitHub angezeigt werden.

Personal access tokens

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](#).

Abbildung 1.7: Generiertes Zugriffstoken

- 1.7. Führen Sie auf `workstation` den Befehl `git config` mit den Parametern `credential.helper cache` aus, um Ihre Anmeldedaten für die spätere Verwendung im Cache-Speicher zu speichern. Der Parameter `--global` wendet die Konfiguration auf alle Ihre Repositorys an.

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



Wichtig

Wenn Sie in diesem Kurs bei der Verwendung von Git-Vorgängen in der Befehlszeile zur Eingabe eines Passworts aufgefordert werden, verwenden Sie Ihr Zugriffstoken als Passwort.

► 2. Bereiten Sie Ihr Quay.io-Passwort vor.

- 2.1. Konfigurieren Sie ein Passwort für Ihr Quay.io-Benutzerkonto. Klicken Sie auf der Seite *Account Settings* auf den Link *Change password*. Weitere Einzelheiten finden Sie in *Anhang C, Erstellen eines Quay-Benutzerkontos*.

► 3. Konfigurieren Sie den Rechner **workstation**.

Verwenden Sie für die folgenden Schritte die Werte, die von der Red Hat Training Online Learning-Umgebung bei der Einrichtung Ihrer Online-Lab-Umgebung bereitgestellt werden:

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

workstation	active	ACTION	OPEN CONSOLE
classroom	active	ACTION	OPEN CONSOLE

Öffnen Sie auf dem Rechner **workstation** ein Terminalfenster, und führen Sie den folgenden Befehl aus. Beantworten Sie die interaktiven Eingabeaufforderungen, bevor Sie eine andere Übung in diesem Kurs beginnen.

Wenn Sie einen Fehler machen, können Sie den Befehl jederzeit mit Strg+C unterbrechen und von vorne beginnen.

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

- 3.1. Der Befehl **lab-configure** startet mit der Anzeige einer Reihe interaktiver Eingabeaufforderungen und nutzt sinnvolle Standardwerte, sofern diese verfügbar sind.

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

- Enter the API Endpoint: `https://api._cluster.domain.example.com_:6443` 1
- Enter the Username: `__youruser__` 2
- Enter the Password: `__yourpassword__` 3
- Enter the GitHub Account Name: `__yourgituser__` 4
- Enter the Quay.io Account Name: `__yourquayuser__` 5

...output omitted...

- ① Die URL zur Master-API Ihres OpenShift-Clusters. Geben Sie die URL als eine einzelne Zeile ohne Leerzeichen oder Zeilenumbrüche ein. Red Hat Training stellt diese Informationen bereit, wenn Sie Ihre Lab-Umgebung einrichten. Sie benötigen diese Informationen, um sich beim Cluster anzumelden sowie um Container-Anwendungen bereitzustellen.
- ② ③ Ihr OpenShift-Entwicklerbenutzername und das zugehörige Passwort. Red Hat Training stellt diese Informationen bereit, wenn Sie Ihre Lab-Umgebung einrichten. Sie müssen diesen Benutzernamen und das Passwort verwenden, um sich bei OpenShift anzumelden. Sie verwenden Ihren Benutzernamen auch als Teil von Bezeichnern wie Routen-Hostnamen und Projektnamen, um Kollisionen mit Bezeichnern anderer Kursteilnehmer zu vermeiden, die denselben OpenShift-Cluster wie Sie nutzen.
- ④ ⑤ Die Namen Ihrer persönlichen GitHub- und Quay.io-Benutzerkonten. Sie benötigen gültige, kostenlose Benutzerkonten für diese Online-Services, um die Übungen dieses Kurses durchzuführen. Wenn Sie noch keinen dieser Online-Services verwendet haben, finden Sie in *Anhang B, Erstellen eines GitHub-Benutzerkontos* und *Anhang C, Erstellen eines Quay-Benutzerkontos* Informationen zur Registrierung.

- 3.2. Mit dem Befehl `lab-configure` werden alle von Ihnen eingegebenen Informationen ausgegeben und es wird versucht, eine Verbindung zu Ihrem OpenShift-Cluster herzustellen.

...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. Wenn `lab-configure` ein Problem findet, wird eine Fehlermeldung angezeigt und der Befehl beendet. Sie müssen Ihre Angaben überprüfen und den Befehl `lab-configure` erneut ausführen. Die folgende Auflistung zeigt ein Beispiel für einen Verifizierungsfehler.

```
...output omitted...

Verifying your API Endpoint...

ERROR:
Cannot connect to an OpenShift 4.6 API using your URL.
Please verify your network connectivity and that the URL does not point to an
OpenShift 3.x nor to a non-OpenShift Kubernetes API.
```

- 3.4. Wenn soweit alles in Ordnung ist, versucht `lab-configure` auf Ihre öffentlichen GitHub- und Quay.io-Benutzerkonten zuzugreifen.

```
...output omitted...

Verifying your GitHub account name...

Verifying your Quay.io account name...

...output omitted...
```

- 3.5. Wenn Probleme gefunden werden, zeigt `lab-configure` eine Fehlermeldung an und wird beendet. Sie müssen Ihre Angaben überprüfen und den Befehl `lab-configure` erneut ausführen. Die folgende Auflistung zeigt ein Beispiel für einen Verifizierungsfehler:

```
...output omitted...

Verifying your GitHub account name...

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

- 3.6. Schließlich überprüft der Befehl `lab-configure`, ob Ihr OpenShift-Cluster die erwartete Platzhalter-Domain ausgibt.

```
...output omitted...

Verifying your cluster configuration...

...output omitted...
```

- 3.7. Wenn alle Prüfungen erfolgreich sind, speichert der Befehl `lab-configure` Ihre Konfiguration:

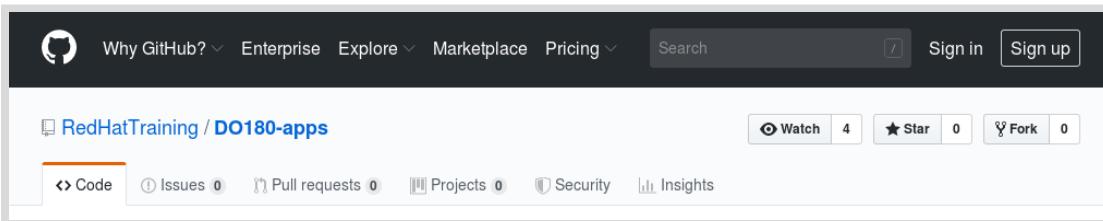
```
...output omitted...

Saving your lab configuration file...

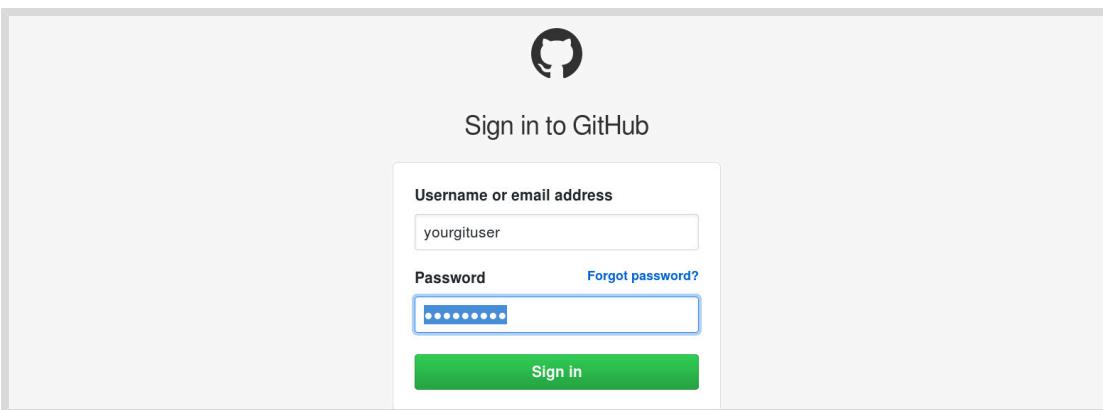
All fine, lab config saved. You can now proceed with your exercises.

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

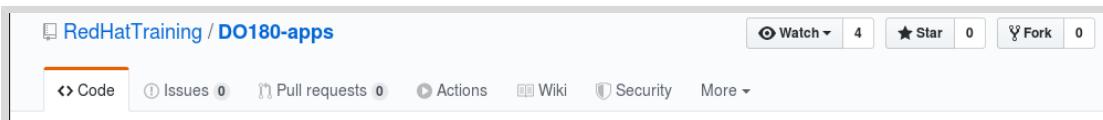
- 3.8. Wenn keine Fehler beim Speichern Ihrer Konfiguration vorliegen, sind Sie fast bereit, eine der Übungen dieses Kurses zu beginnen. Falls Fehler auftreten, versuchen Sie nicht, eine Übung zu beginnen, bevor Sie den Befehl `lab-configure` erfolgreich ausführen können.
- 4. Verzweigen Sie die Beispielanwendungen des Kurses auf Ihr persönliches GitHub-Benutzerkonto. Führen Sie die folgenden Schritte aus:
- 4.1. Öffnen Sie einen Webbrower und navigieren Sie zu <https://github.com/RedHatTraining/DO180-apps>. Wenn Sie nicht bei GitHub angemeldet sind, klicken Sie in der rechten oberen Ecke auf Sign in.



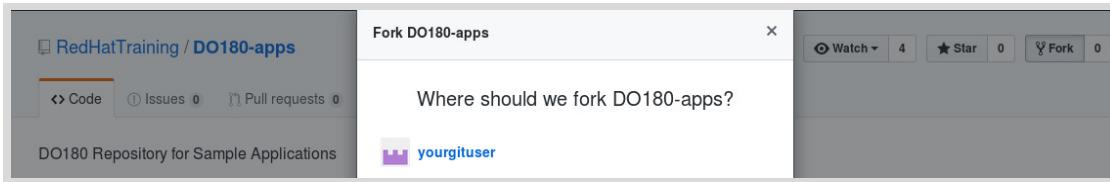
- 4.2. Melden Sie sich bei GitHub mit Ihrem persönlichen Benutzernamen und dem Passwort an.



- 4.3. Navigieren Sie zum Repository RedHatTraining/DO180-apps, und klicken Sie in der rechten oberen Ecke auf Fork.



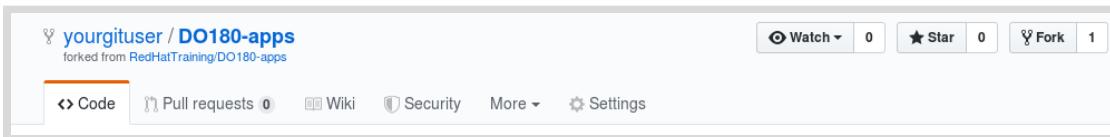
- 4.4. Klicken Sie im Fenster Fork DO180-apps auf yourgituser, um Ihr persönliches GitHub-Projekt auszuwählen.



Wichtig

Es ist zwar möglich, Ihren persönlichen Branch des Repositorys <https://github.com/RedHatTraining/DO180-apps> umzubenennen, aber für die Auswertungsskripts, Hilfsskripts und die Beispieldaten in diesem Kurs ist es erforderlich, dass Sie beim Verzweigen des Repositorys den Namen **D0180-apps** beibehalten.

- 4.5. Nach ein paar Minuten wird in der GitHub-Weboberfläche Ihr neues *yourgituser/D0180-apps*-Repository angezeigt.



- ▶ 5. Klonen Sie diese Beispieldaten des Kurses von Ihrem persönlichen GitHub-Benutzerkonto auf Ihren **workstation**-Rechner. Führen Sie die folgenden Schritte aus:

- 5.1. Führen Sie den folgenden Befehl aus, um das Beispieldaten-Repository des Kurses zu klonen. Ersetzen Sie *yourgituser* durch den Namen Ihres persönlichen GitHub-Benutzerkontos.

```
[student@workstation ~]$ git clone https://github.com/_yourgituser/_D0180-apps
Cloning into 'D0180-apps'...
...output omitted...
```

- 5.2. Überprüfen Sie, ob es sich bei `/home/user/D0180-apps` um ein Git-Repository handelt.

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

- 5.3. Erstellen Sie einen neuen Branch, um Ihr neues persönliches Zugriffstoken zu testen.

```
[student@workstation D0180-apps]$ git checkout -b testbranch
Switched to a new branch `testbranch`
```

- 5.4. Nehmen Sie eine Änderung an der Datei TEST vor, und übergeben Sie sie dann an Git.

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

- 5.5. Übertragen Sie die Änderungen an Ihren kürzlich erstellten Test-Branch.

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

- ① Geben Sie Ihren GitHub-Benutzernamen ein.
 - ② Geben Sie Ihr persönliches Zugriffstoken ein.
- 5.6. Nehmen Sie andere Änderungen an einer Textdatei vor, übergeben und übertragen Sie sie. Sie werden feststellen, dass Sie nicht mehr aufgefordert werden, Ihren Benutzer und Ihr Passwort einzugeben. Das liegt an dem Befehl `git config`, den Sie in Schritt 7 ausgeführt haben.

```
[student@workstation D0180-apps]$ echo "OCP4.6" > TEST
[student@workstation D0180-apps]$ git add .
[student@workstation D0180-apps]$ git commit -am "OCP4.6"
[student@workstation D0180-apps]$ git push
...output omitted...
```

- 5.7. Überprüfen Sie, ob `/home/user/D0180-apps` die Beispieldaten dieses Kurses enthält, und wechseln Sie zurück in den Benutzerordner des Benutzers.

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

Nachdem Sie nun einen lokalen Klon des `D0180-apps`-Repositorys auf Ihrem Rechner `workstation` erstellt und den Befehl `lab-configure` erfolgreich ausgeführt haben, können Sie die Übungen dieses Kurses beginnen.

In diesem Kurs starten alle Übungen, die Anwendungen aus Quellcode erstellen, aus dem Branch `master` des Git-Repositorys `D0180-apps`. Bei Übungen, die Änderungen am Quellcode vornehmen, müssen Sie neue Branches erstellen, um Ihre Änderungen zu hosten, sodass der `master`-Branch immer einen als fehlerfrei bekannten Startpunkt enthält. Wenn Sie aus irgendeinem Grund eine Übung unterbrechen oder neu starten müssen und die Änderungen, die Sie in Ihren Git-Branches vornehmen, entweder speichern oder verwerfen müssen, finden Sie weitere Informationen unter *Anhang E, Nützliche Git-Befehle*.

Hiermit ist die angeleitete Übung beendet.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Container stellen eine isolierte Anwendungslaufzeit dar, die mit geringem Overhead erstellt wird.
- Container-Images verpacken Anwendungen mit allen zugehörigen Abhängigkeiten. Dies vereinfacht die Ausführung der Anwendung in verschiedenen Umgebungen.
- Anwendungen wie Podman erstellen Container mithilfe von Features des Linux-Standardkernels.
- Container-Image-Registries werden bevorzugt zur Verteilung von Container-Images an mehrere Benutzer und Hosts verwendet.
- OpenShift organisiert mithilfe von Kubernetes Anwendungen, die aus mehreren Containern bestehen.
- Kubernetes verwaltet Load Balancing, Hochverfügbarkeit und den persistenten Storage für containerisierte Anwendungen.
- OpenShift ergänzt die Funktionen von Kubernetes zur Multi-Tenancy, Sicherheit, Benutzerfreundlichkeit und für Continuous Integration und Continuous Development.
- OpenShift-Routen ermöglichen den externen Zugriff auf containerisierte Anwendungen in überschaubarer Weise.

Kapitel 2

Erstellen von containerisierten Services

Ziel

Bereitstellen eines Services mit Container-Technologie

Ziele

- Erstellen eines Datenbankservers anhand eines Container-Images

Abschnitte

- Bereitstellen eines containerisierten Datenbankservers (und angeleitete Übung)
- Verwenden von Containern (und angeleitete Übung)

Praktische Übung

- Erstellen von containerisierten Services

Bereitstellen von containerisierten Services

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Suchen und Abrufen von Container-Images mit Podman
- Lokales Ausführen und Konfigurieren von Containern
- Verwenden von Red Hat Container Catalog

Abrufen von Container-Images mit Podman

Anwendungen können in Containern ausgeführt werden, um eine isolierte und kontrollierte Ausführungsumgebung zu bieten. Zum Ausführen einer containerisierten Anwendung, d. h. zum Ausführen einer Anwendung in einem Container, sind ein Container-Image, ein Dateisystempaket, das alle Anwendungsdateien enthält, Librarys und Abhängigkeiten, welche die Anwendung zur Ausführung benötigt, erforderlich. Container-Images befinden sich in Image-Registries, mit denen Benutzer Container-Images suchen und abrufen können. Podman-Benutzer können den Unterbefehl `search` ausführen, um in Remote- oder lokalen Registries nach verfügbaren Images zu suchen.

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

Nachdem Sie ein Image gefunden haben, können Sie es mit Podman herunterladen. Führen Sie den Unterbefehl `pull` aus, um Podman anzuweisen, das Image abzurufen und es zur späteren Verwendung lokal zu speichern.

```
[user@demo ~]$ podman pull rhe1
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
```

Container-Images werden anhand der folgenden Syntax benannt:

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

Registry-Benennungssyntax:

- Bei `registry_name` handelt es sich um den Namen der das Image speichernden Registry. Hierbei handelt es sich normalerweise um den vollqualifizierten Domain-Namen der Registry.
- Der `user_name` ist der Name des Benutzers oder der Organisation, zu dem bzw. der das Image gehört.
- Der `image_name` muss im Benutzer-Namespace eindeutig sein.
- Das Tag gibt die Image-Version an. Wenn der Image-Name kein Image-Tag enthält, wird `latest` angenommen.



Anmerkung

Bei der Podman-Installation dieses Kursraums werden mehrere öffentlich zugängliche Registries verwendet, etwa Quay .io und Red Hat Container Catalog.

Nach dem Abruf speichert Podman die Images lokal, und Sie können sie mit dem Unterbefehl `images` auflisten:

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

Ausführen von Containern

Der Befehl `podman run` führt einen Container lokal basierend auf einem Image aus. Für den Befehl ist mindestens der Name des Images erforderlich, um im Container ausgeführt zu werden.

Das Container-Image gibt einen Prozess an, der innerhalb des Containers gestartet und als Einstiegspunkt bezeichnet wird. Der Befehl `podman run` verwendet alle Parameter nach dem Image-Namen als Einstiegspunktbefehl für den Container. Im folgenden Beispiel wird ein Container aus einem Red Hat Universal Base Image gestartet. Der Einstiegspunkt für diesen Container wird auf den Befehl `echo "Hello world"` festgelegt.

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

Geben Sie die Option `-d` an den Befehl `podman run` weiter, um ein Container-Image als einen Hintergrundprozess zu starten:

```
[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...
```

In diesem Beispiel wird ein Container-Apache-HTTP-Server im Hintergrund ausgeführt. Er verwendet die Option `-p 8080`, um den Port des HTTP-Servers an einen lokalen Port zu binden. Anschließend wird mit dem Befehl `podman port` der lokale Port abgerufen, den der Container überwacht. Abschließend verwendet er diesen Port, um die Ziel-URL zu erstellen und die Root-Seite vom Apache-HTTP-Server abzurufen. Diese Antwort beweist, dass der Container nach der Ausführung des Befehls `podman run` weiterhin ausgeführt wird.



Anmerkung

Die meisten Unterbefehle von Podman akzeptieren das Flag `-l` (1 für „latest“) als einen Ersatz für die Container-ID. Dieses Flag wendet den Befehl auf den zuletzt verwendeten Container an, der in einem beliebigen Podman-Befehl verwendet wurde.



Anmerkung

Wenn das auszuführende Image beim Ausführen des Befehls `podman run` lokal nicht verfügbar ist, verwendet Podman automatisch `pull`, um das Image herunterzuladen.

Beim Verweisen auf den Container erkennt Podman einen Container entweder anhand des Containernamens oder der generierten Container-ID. Verwenden Sie die Option `--name`, um den Container-Namen beim Ausführen des Containers mit Podman festzulegen. Container-Namen müssen eindeutig sein. Wenn im Befehl `podman run` kein Container-Name enthalten ist, generiert Podman einen eindeutigen Zufallsnamen für Sie.

Wenn für die Images eine Interaktion des Benutzers mit der Konsole erforderlich ist, kann Podman die Eingabe- und Ausgabeströme von Containern an die Konsole umleiten. Zum Aktivieren der Interaktivität schreibt der Unterbefehl `run` die Flags `-t` und `-i` (oder kurz gesagt das Flag `-it`) vor.



Anmerkung

Viele Podman-Flags haben auch eine alternative Langform. Einige davon werden im Folgenden erklärt:

- `-t` entspricht `--tty`, d. h., für den Container muss `pseudo-tty` (Pseudo-Terminal) zugeordnet werden.
- `-i` entspricht `--interactive`. Bei Verwendung wird die Standardeingabe in den Container geöffnet gehalten.
- Bei `-d` oder der zugehörigen Langform `--detach` wird der Container im Hintergrund (getrennt) ausgeführt. Podman gibt dann die Container-ID aus.

Die vollständige Liste der Flags finden Sie in der Podman-Dokumentation.

Das folgende Beispiel startet *innerhalb* des Containers ein Bash-Terminal und führt darin interaktiv einige Befehle aus:

```
[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 ~]$
```

Einige Container müssen oder können externe Parameter verwenden, die beim Start angegeben werden. Der gebräuchlichste Ansatz zum Bereitstellen und Verwenden dieser Parameter erfolgt über Umgebungsvariablen. Podman kann beim Start Umgebungsvariablen in Container einschleusen, indem dem Unterbefehl `run` das Flag `-e` hinzugefügt wird.

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

Im vorherigen Beispiel wird ein UBI-Image-Container gestartet, in dem die beiden als Parameter angegebenen Umgebungsvariablen ausgegeben werden.

Ein weiterer Anwendungsfall für Umgebungsvariablen ist das Einrichten von Anmeldeinformationen in einem MySQL-Datenbankserver.

```
[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
```

Verwenden von Red Hat Container Catalog

Von Red Hat wird ein Repository genau angepasster Container-Images verwaltet. Die Verwendung dieses Repositorys bietet einen zuverlässigen Schutz gegen bekannte Schwachstellen, die durch nicht getestete Images entstehen können. Der Standardbefehl `podman` ist mit Red Hat Container Catalog kompatibel. Der Red Hat Container Catalog bietet eine benutzerfreundliche Oberfläche zum Suchen und Untersuchen von Container-Images aus dem Red Hat-Repository.

Der Container Catalog ist zudem die zentrale Schnittstelle für den Zugriff auf die verschiedenen Aspekte aller im Repository verfügbaren Container-Images. Das ist nützlich, um anhand der Zustandsindexstufen das beste Image von mehreren Versionen eines Container-Images zu bestimmen. Mit der Zustandsindexstufe wird angegeben, wie aktuell ein Image ist und ob es die neuesten Sicherheitsaktualisierungen enthält.

Über den Container Catalog kann zudem auf die Errata-Dokumentation für ein Image zugegriffen werden. Dort werden die neuesten Fehlerkorrekturen und Erweiterungen jedes Updates beschrieben. Zudem wird dort das beste Verfahren zum Abrufen eines Images unter den einzelnen Betriebssystemen angegeben.

Auf den folgenden Bildern sind einige der Red Hat Container Catalog-Features zu sehen.

The screenshot shows the Red Hat Container Catalog search interface. A search bar at the top contains the query 'Apache httpd'. Below the search bar, there are three filter categories: 'Provider' (IBM, Red Hat, Inc.), 'Category' (Database & Data Management, Programming Languages & Runtimes, Web Services), and 'Product'. The search results display three container images from Red Hat:

- rhel8/httpd-24** (Apache httpd 2.4) by Red Hat, Inc. (Updated 11 hours ago)
- rhel8/httpd-24** (Apache httpd 2.4) by Red Hat, Inc. (Updated 11 hours ago)
- ibm/couchdb3** (Apache CouchDB) by IBM (Updated 4 days ago)

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

Abbildung 2.1: Suchseite von Red Hat Container Catalog

Wie im vorherigen Image gezeigt, wird durch Eingabe von Apache httpd in das Container Catalog-Suchfeld eine Liste mit Vorschlägen zu Produkten und Image-Repositorys angezeigt, die zur Suche passen. Um die Apache httpd 2.4-Image-Seite zu öffnen, wählen Sie in der Liste rhel8/httpd-24 aus.

Nach Auswahl des gewünschten Images enthält die nachfolgende Seite zusätzliche Informationen zum Image:

The screenshot shows the detailed view of the Apache httpd 2.4 image. At the top, it displays the image name 'Apache httpd 2.4' and its provider 'Red Hat'. Below this, there are dropdown menus for 'Architecture' (amd64) and 'Tag' (latest). The main content area includes tabs for 'Overview', 'Security', 'Packages', and 'Dockerfile'. The 'Overview' tab is selected. It contains a brief description of the Apache HTTP Server, a 'Get this image' button, and a 'Description' section. The 'Description' section includes a detailed paragraph about the server's features and capabilities. To the right of the description, there are sections for 'Published' (28 days ago), 'Release category' (Generally Available), 'Health index' (B 2), and a 'Have feedback?' button. At the bottom, there is a note about usage in OpenShift and a link to the imagestream tag in Openshift.

Abbildung 2.2: Apache httpd 2.4-Image-Übersichtsseite (rhel8/httpd-24)

Im Bereich *Apache httpd 2.4* werden Image-Details und mehrere Registerkarten angezeigt. Auf dieser Seite ist angegeben, dass das Image-Repository von Red Hat verwaltet wird.

Kapitel 2 | Erstellen von containerisierten Services

Unter der Registerkarte Overview sind weitere Details ausgeführt:

- *Description*: Eine Zusammenfassung der Funktionen des Images.
- *Documentation*: Verweise auf die Autorendokumentation des Containers.
- *Products using this container*: Gibt an, dass Red Hat Enterprise Linux dieses Image-Repository verwendet.

Auf der rechten Seite werden Informationen dazu angezeigt, wann das Image das letzte Update erhalten hat, welches Tag zuletzt auf das Image angewendet wurde, seinen Gesundheitszustand, seine Größe und mehr.

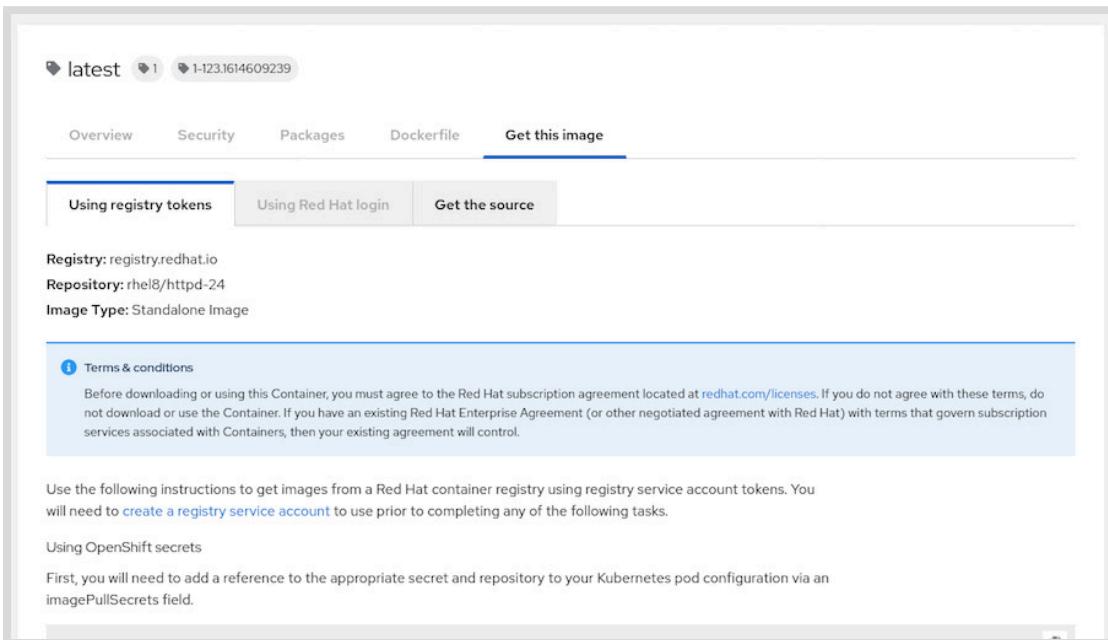


Abbildung 2.3: Apache httpd 2.4-Seite mit dem neuesten Image (rhel8/httpd-24)

Auf der Registerkarte *Get this image* wird das Verfahren zum Abrufen der neuesten Version des Images beschrieben. Auf der Seite sind verschiedene Optionen zum Abrufen des Images enthalten. Wählen Sie die gewünschte Prozedur auf den Registerkarten aus, und die Seite zeigt die entsprechenden Anweisungen zum Abrufen des Images an.

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

Quay.io-Website
<https://quay.io>

► Angeleitete Übung

Erstellen einer MySQL-Datenbankinstanz

In dieser Übung starten Sie eine MySQL-Datenbank in einem Container und erstellen dann eine Datenbank und füllen sie auf.

Ergebnisse

Sie sollten in der Lage sein, eine Datenbank in einem Container-Image zu starten und Informationen in der Datenbank zu speichern.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer student auf workstation ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

► 1. Erstellen Sie eine MySQL-Containerinstanz.

- 1.1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an. Lesen Sie die Anweisungen in *Anhang D, Erstellen eines Red Hat-Benutzerkontos*, wenn Sie sich bei Red Hat registrieren müssen.

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

- 1.2. Starten Sie einen Container über das MySQL-Image von 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
```

Mit diesem Befehl wird das MySQL 8.0-Container-Image mit dem Tag 1 heruntergeladen, und ein Container wird basierend auf diesem Image gestartet. Damit wird eine Datenbank namens `items` erstellt, deren Besitzer der Benutzer `user1` mit `mypa55` als Passwort ist. Das Datenbankadministratorpasswort ist auf `r00tpa55` festgelegt, und der Container wird im Hintergrund ausgeführt.

- 1.3. Verifizieren Sie, ob der Container fehlerfrei gestartet wurde.

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

- 2. Greifen Sie durch Ausführen des folgenden Befehls auf die Container-Sandbox zu:

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

Der Befehl startet eine Bash-Shell, die im MySQL-Container als Benutzer `mysql` ausgeführt wird:

- 3. Fügen Sie der Datenbank Daten hinzu.

- 3.1. Melden Sie sich bei MySQL als der Datenbankadministrator (root) an.

Führen Sie im Container-Terminal den folgenden Befehl aus, um eine Verbindung zur Datenbank herzustellen:

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

Mit dem Befehl `mysql` wird die interaktive Befehlszeile für die MySQL-Datenbank geöffnet. Führen Sie den folgenden Befehl aus, um die Datenbankverfügbarkeit zu ermitteln:

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

- 3.2. Erstellen Sie in der Datenbank `items` eine neue Tabelle. Führen Sie den folgenden Befehl aus, um auf die Datenbank zuzugreifen.

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

- 3.3. Erstellen Sie in der Datenbank `items` eine Tabelle mit dem Namen `Projects`.

```
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)
```

Kapitel 2 | Erstellen von containerisierten Services

Sie können optional die Datei `~/D0180/solutions/container-create/create_table.txt` verwenden, um die bereitgestellte MySQL-Anweisung `CREATE TABLE` zu kopieren und einzufügen.

- 3.4. Verwenden Sie den Befehl `show tables`, um zu überprüfen, ob die Tabelle erstellt wurde.

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

- 3.5. Verwenden Sie den Befehl `insert`, um eine Zeile in die Tabelle einzufügen.

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

- 3.6. Verwenden Sie den Befehl `select`, um zu überprüfen, ob der Tabelle die Projektinformationen hinzugefügt wurden.

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

- 3.7. Beenden Sie die MySQL-Befehlszeile und den MySQL-Container.

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

Beenden

Führen Sie auf `workstation` das Skript `lab container-create finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit ist diese Übung beendet.

Verwenden von rootless-Containern

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Erläutern der Unterschiede zwischen dem Ausführen von root- und rootless-Containern
- Beschreiben der Vor- und Nachteile der einzelnen Fälle
- Ausführen als root- und rootless-Container mit Podman

Weiterentwicklung der Container-Nutzung

Wenn Sie Container seit einiger Zeit ausführen, führen Sie sie wahrscheinlich als privilegierter Benutzer aus. In der Vergangenheit war es bei der Erstellung von Containern erforderlich, dass Laufzeit-Engines als root ausgeführt werden, und privilegierter Zugriff war erforderlich, um Ressourcen wie Netzwerkschnittstellen zu erstellen.

Aus Sicherheitssicht empfiehlt es sich nicht, diese Zugriffsebene bereitzustellen. Sie sollten Software immer mit Berechtigungen ausführen, die möglichst eingeschränkt sind. Wenn ein Sicherheitsfehler entweder auf der Laufzeit-Engine oder in der Anwendung selbst ausgenutzt wird, werden dadurch die Auswirkungen minimiert.

Eine bessere Vorgehensweise besteht in der Containerisierung von Anwendungen, sodass für die Ausführung kein privilegierter Benutzer erforderlich ist. Diese Anwendungen sollten stattdessen einen bekannten Benutzer verwenden.

Für viele Community-Container-Images, beispielsweise die, die unter docker.io verfügbar sind, muss weiterhin root ausgeführt werden.

Einige Tools, wie Podman und Red Hat OpenShift, führen standardmäßig rootless-Container aus. Docker hat angekündigt, dass der rootless-Modus in der Version 20.10 allgemein verfügbar ist.

Vorteile von rootless-Containern

Rootless-Container sind ein neues Konzept von Containern, für deren Ausführung keine Root-Berechtigungen erforderlich sind. Rootless-Container sind aus Sicherheitssicht aus verschiedenen Gründen vorteilhaft, darunter die folgenden:

- Ermöglichung der Ausführung von Code in einem rootless-Container mit root-Berechtigungen, ohne dass er als root-Benutzer des Hosts ausgeführt werden muss.
- Hinzufügen einer neuen Sicherheitsschicht. Wenn die Container-Engine kompromittiert ist, erhält der Angreifer keine Root-Berechtigungen auf dem Host.
- Ermöglicht es mehreren unprivilegierten Benutzern, Container auf demselben Rechner auszuführen.
- Ermöglicht die Isolierung innerhalb verschachtelter Container.

Trotz all dieser Vorteile gelten für die Ausführung von rootless-Containern verschiedene Einschränkungen, darunter die folgenden:

Kapitel 2 | Erstellen von containerisierten Services

- Entfernte Funktionen
- Bindung an Ports unter 1024
- Volume, das andere Inhalte bereitstellt

Eine detaillierte Liste der Mängel von rootless finden Sie <https://github.com/containers/podman/blob/master/rootless.md>.

Grundlagen von rootless-Containern

Um zu verstehen, wie rootless-Container funktionieren, sollten Sie die folgenden Konzepte beachten.

Benutzer-Namespaces

Container verwenden Linux-Namespace, um sich vom Host zu isolieren, auf dem sie ausgeführt werden. Insbesondere wird der Benutzer-Namespace verwendet, um Container zu rootless-Containern zu machen. Dieser Namespace ordnet Benutzer- und Gruppen-IDs zu, sodass ein Prozess im Namespace möglicherweise unter einer anderen ID ausgeführt zu werden scheint.

Rootless-Container verwenden den Benutzer-Namespace, damit Anwendungscode scheinbar als root ausgeführt wird. Aus Sicht des Hosts sind die Berechtigungen jedoch auf die eines regulären Benutzers beschränkt. Wenn ein Angreifer den Benutzer-Namespace auf dem Host verlassen kann, verfügt er nur über die Funktionen eines regulären, unprivilegierten Benutzers.

Netzwerkfunktionen

Um ordnungsgemäße Netzwerkfunktionen innerhalb eines Containers zu ermöglichen, wird ein virtuelles Ethernet-Gerät erstellt. Dies stellt ein Problem für rootless-Container dar, da nur ein echter root-Benutzer über die Berechtigungen verfügt, um dieses und ähnliche Geräte zu erstellen.

In einem rootless-Container werden die Netzwerkfunktionen in der Regel von Slirp verwaltet. Dabei wird in die Benutzer- und Netzwerk-Namespace des Containers verzweigt, und es wird ein TAP-Gerät erstellt, das zur Standardroute wird. Anschließend wird der Dateideskriptor des Geräts an das übergeordnete Element übertragen, das im standardmäßigen Netzwerk-Namespace ausgeführt wird und nun sowohl mit dem Container als auch mit dem Internet kommunizieren kann.

Storage

Container-Engines verwenden standardmäßig einen speziellen Treiber namens Overlay2 (oder Overlay), um ein mehrschichtiges Dateisystem zu erstellen, das sowohl hinsichtlich Kapazität als auch Leistung effizient ist. Dies kann nicht mit rootless-Containern erfolgen, da die meisten Linux-Distributionen das Mounten von Overlay-Dateisystemen in Benutzer-Namespace nicht zulassen.

Bei rootless-Containern besteht die Lösung in der Erstellung eines neuen Storage-Treibers. FUSE-OverlayFS ist eine Benutzbereichsimplementation von Overlay, das effizienter ist als der zuvor verwendete VFS-Storage-Treiber und in Benutzer-Namespace ausgeführt werden kann.



Literaturhinweise

user_namespaces(7)-Manpage

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

Verwendung von Podman in einer rootless-Umgebung

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

► Angeleitete Übung

Erkunden von root- und rootless-Containern

In dieser Übung werden Sie die Unterschiede zwischen der Ausführung von Containern als root und ihrer Ausführung als rootless verstehen.

Ergebnisse

Sie sollten in der Lage sein, die UIDs für Prozesse anzuzeigen, die in Containern ausgeführt werden.

Bevor Sie Beginnen

Führen Sie auf dem Rechner `workstation` als Benutzer `student` den Befehl `lab` aus, um Ihr System für diese Übung vorzubereiten.

Mit diesem Befehl wird sichergestellt, dass die zum Durchführen dieser Übung erforderlichen Tools verfügbar sind.

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

Anweisungen

- ▶ 1. Führen Sie einen Container als root-Benutzer aus, und überprüfen Sie die UID eines darin ausgeführten Prozesses.
 - 1.1. Starten Sie einen Container mit sudo aus dem Red Hat UBI 8-Image.

```
[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. Starten Sie einen `sleep`-Prozess im Container.

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

- 1.3. Führen Sie auf einem neuen Terminal `ps` aus, um nach dem Prozess zu suchen.

```
[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. Beenden Sie den Container.

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

- 2. Führen Sie einen Container als regulärer Benutzer aus, und überprüfen Sie die UID eines darin ausgeführten Prozesses.

2.1. Starten Sie einen weiteren Container aus dem Red Hat UBI 8 als regulärer Benutzer.

```
[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. Starten Sie einen sleep-Prozess im Container.

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

2.3. Führen Sie auf einem neuen Terminal ps aus, um nach dem Prozess zu suchen.

```
[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. Beenden Sie den Container.

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

Beenden

Führen Sie auf dem Rechner **workstation** den Befehl **lab** aus, um diese Übung zu beenden. Dies ist wichtig, um sicherzustellen, dass Ressourcen aus vorherigen Übungen sich nicht auf zukünftige Übungen auswirken.

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

Hiermit ist der Abschnitt abgeschlossen.

► Praktische Übung

Erstellen von containerisierten Services

Ergebnisse

Sie sollten in der Lage sein, einen Container mit einem Container-Image zu starten und anzupassen.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer student auf dem Rechner workstation einen Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Verwenden Sie das Image quay.io/redhattraining/httpd-parent mit dem Tag 2.4, um einen neuen Container namens httpd-basic im Hintergrund zu starten. Leiten Sie Port 8080 auf dem Host zu Port 80 im Container weiter.
2. Testen Sie, ob der Apache HTTP-Server im Container httpd-basic ausgeführt wird.
3. Passen Sie den httpd-basic-Container so an, dass Hello World als Meldung angezeigt wird. Die Meldung des Containers wird in der Datei /var/www/html/index.html gespeichert.

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem workstation-Rechner den Befehl lab container-review grade ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf dem Rechner workstation das Skript lab container-review finish aus, um diese praktische Übung zu beenden.

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

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Erstellen von containerisierten Services

Ergebnisse

Sie sollten in der Lage sein, einen Container mit einem Container-Image zu starten und anzupassen.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf dem Rechner `workstation` einen Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Verwenden Sie das Image `quay.io/redhattraining/httpd-parent` mit dem Tag `2.4`, um einen neuen Container namens `httpd-basic` im Hintergrund zu starten. Leiten Sie Port `8080` auf dem Host zu Port `80` im Container weiter.
 - 1.1. Verwenden Sie den Befehl `podman run`, um einen Container zu starten. Fügen Sie die Option `-d` hinzu, um ihn im Hintergrund zu starten, und fügen Sie die Option `-p` hinzu, um Port `8080` auf dem Host Port `80` im Container zuzuordnen.

```
[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` aaf94b3a4a54485d76a0a094cbfac89c287d360890a3d2779a5a
```

Durch diesen Befehl wird Apache HTTP Server im Hintergrund gestartet, und die Bash-Befehlszeile wird zurückgegeben.

2. Testen Sie, ob der Apache HTTP-Server im Container `httpd-basic` ausgeführt wird.
 - 2.1. Versuchen Sie, von dem Rechner `workstation` mit einem beliebigen Webbrowser auf `http://localhost:8080` zuzugreifen.

Die Meldung `Hello from the httpd-parent container!` wird angezeigt. Diese Meldung befindet sich auf der Seite `index.html` im Apache HTTP Server-Container, der auf `workstation` ausgeführt wird.

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

3. Passen Sie den httpd-basic-Container so an, dass Hello World als Meldung angezeigt wird. Die Meldung des Containers wird in der Datei /var/www/html/index.html gespeichert.

- 3.1. Starten Sie innerhalb des Containers eine Bash-Sitzung.

Führen Sie den folgenden Befehl aus:

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

- 3.2. Überprüfen Sie in der Bash-Sitzung die Datei index.html im Verzeichnis /var/www/html mit dem Befehl 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. Ersetzen Sie alle vorhandenen Inhalte der Datei index.html durch „Hello World“.

Führen Sie in der Bash-Sitzung im Container den folgenden Befehl aus:

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

- 3.4. Versuchen Sie, erneut auf http://localhost:8080 zuzugreifen, und überprüfen Sie, ob die Webseite aktualisiert wurde.

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

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem workstation-Rechner den Befehl lab container-review grade ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf dem Rechner workstation das Skript lab container-review finish aus, um diese praktische Übung zu beenden.

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

Hiermit wird die praktische Übung abgeschlossen.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Mit Podman können Benutzer Images von lokalen oder Remote-Registries suchen und herunterladen.
- Der Befehl `podman run` erstellt und startet einen Container über ein Container-Image.
- Container werden mithilfe des Flags `-d` im Hintergrund oder mithilfe des Flags `-it` interaktiv ausgeführt.
- Einige Container-Images erfordern Umgebungsvariablen, die mithilfe der Option `-e` mit dem Befehl `podman run` festgelegt werden.
- Red Hat Container Catalog unterstützt die Suche, Überprüfung und Analyse von Container-Images aus dem offiziellen Container-Image-Repository von Red Hat.

Kapitel 3

Verwalten von Containern

Ziel

Ändern vordefinierter Container-Images zum Erstellen und Verwalten containerisierter Services

Ziele

- Verwalten des Lebenszyklus eines Containers von der Erstellung bis zur Löschung
- Speichern der Container-Anwendungsdaten mit persistentem Storage
- Beschreiben, wie die Portweiterleitung für den Zugriff auf einen Container verwendet wird

Abschnitte

- Verwalten des Lebenszyklus von Containern (und angeleitete Übung)
- Anhängen von persistentem Storage (und angeleitete Übung)
- Zugreifen auf Container (und angeleitete Übung)

Praktische Übung

- Verwalten von Containern

Verwalten des Lebenszyklus von Containern

Ziele

Nach Abschluss dieses Abschnitts sollten Kursteilnehmer den Lebenszyklus eines Containers von der Erstellung bis zur Löschung verwalten können.

Verwalten des Lebenszyklus von Containern mit Podman

In den vorangegangenen Kapiteln haben Sie erfahren, wie Sie Podman verwenden, um einen Container-Service zu erstellen. Im Folgenden werden die Befehle und Strategien ausführlicher beschrieben, mit denen Sie den Lebenszyklus eines Containers verwalten können. Mit Podman können Sie nicht nur Container ausführen, sondern diese auch im Hintergrund ausführen, neue Prozesse in ihnen ausführen und ihnen Ressourcen wie Dateisystem-Volumes oder ein Netzwerk bereitstellen.

Der mit dem Befehl `podman` implementierte Podman bietet eine Reihe von Unterbefehlen zum Erstellen und Verwalten von Containern. Entwickler verwenden diese Unterbefehle, um den Lebenszyklus von Containern und Containerbildern zu verwalten. In der folgenden Abbildung wird eine Zusammenfassung der am häufigsten genutzten Unterbefehle angezeigt, die den Container- und Image-Status ändern:

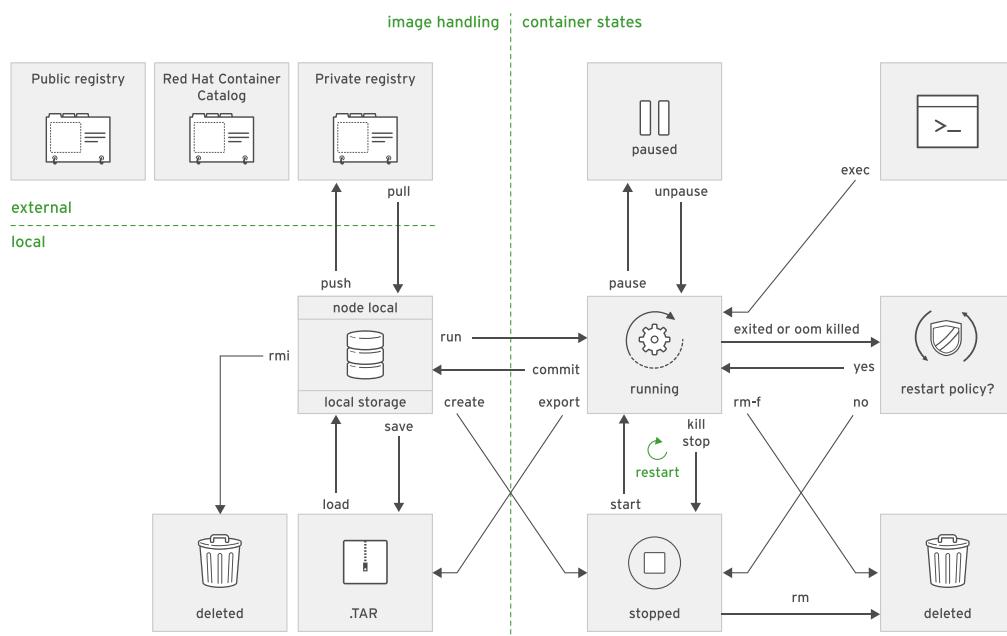


Abbildung 3.1: Podman verwaltet Unterbefehle

Podman bietet darüber hinaus eine Reihe nützlicher Unterbefehle zum Abrufen von Informationen über ausgeführte und beendete Container.

Mit diesen Unterbefehlen können Sie Informationen aus Containern und Images für Debugging-, Aktualisierungs- oder Berichterstellungszwecke extrahieren. In der folgenden Abbildung wird eine

Zusammenfassung der am häufigsten genutzten Unterbefehle angezeigt, die Informationen von Containern und Images abfragen:

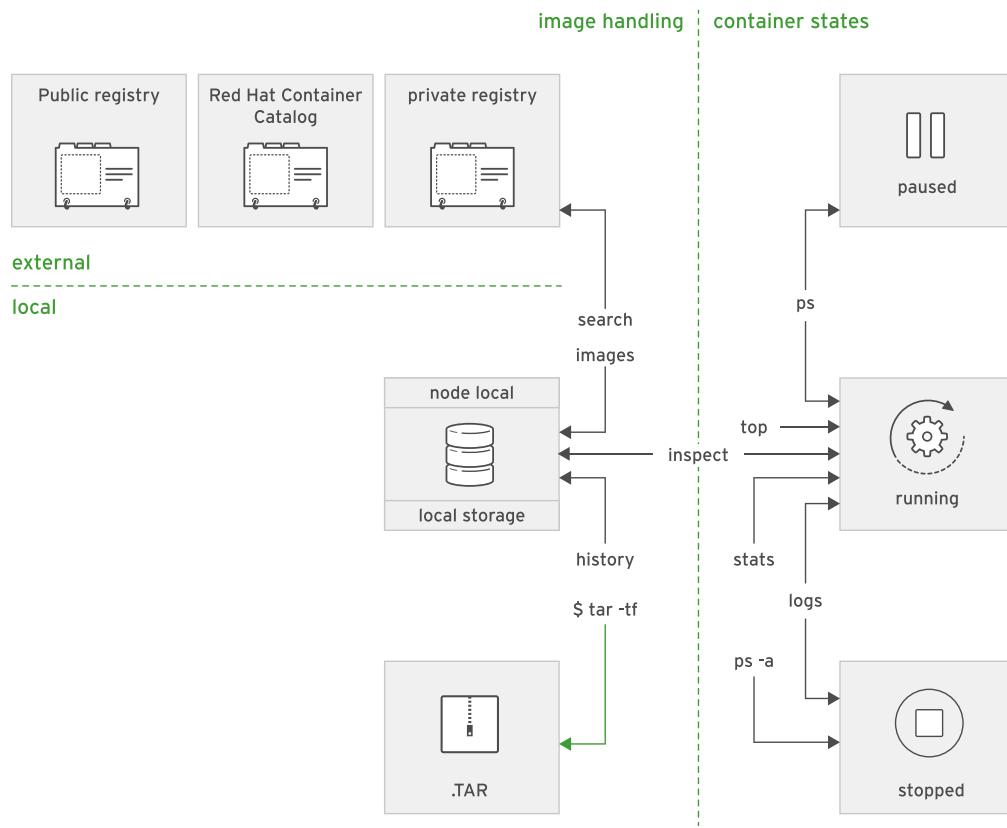


Abbildung 3.2: Unterbefehle für die Podman-Abfrage

Während Sie sich mit den Podman-Sub-Befehlen in diesem Kurs vertraut machen, sollten Sie diese zwei Abbildungen als Referenz verwenden.

Erstellen von Containern

Mit dem Befehl `podman run` wird ein neuer Container über ein Image erstellt und ein Prozess im neuen Container gestartet. Wenn das Container-Image nicht lokal verfügbar ist, versucht dieser Befehl, das Image mithilfe des konfigurierten Image-Repositorys herunterzuladen:

```
[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
```

Im diesem Ausgabebeispiel wurde der Container mit einem nicht interaktiven Prozess gestartet (ohne die Option `-it`) und wird im Vordergrund ausgeführt, da er nicht mit der Option `-d` gestartet wurde. Wenn der resultierende Prozess mit Strg+C (SIGINT) gestoppt wird, werden der Container-Prozess und der Container selbst gestoppt.

Kapitel 3 | Verwalten von Containern

Podman identifiziert Container anhand einer eindeutigen Container-ID oder eines Container-Namens. Der Befehl `podman ps` zeigt die Container-ID und -Namen für alle aktiv ausgeführten Container an:

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

- ① Die Container-ID ist eindeutig und wird automatisch generiert.
- ② Der Container-Name kann manuell angegeben werden, andernfalls wird er automatisch generiert. Dieser Name muss eindeutig sein. Andernfalls schlägt der Befehl `run` fehl.

Der Befehl `podman run` generiert automatisch eine eindeutige Zufalls-ID. Darüber hinaus generiert er einen zufälligen Container-Namen. Verwenden Sie beim Ausführen eines Containers die Option `--name`, um den Container-Namen explizit zu definieren:

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



Anmerkung

Der Name muss eindeutig sein. Podman gibt einen Fehler aus, wenn der Name bereits von einem Container verwendet wird, einschließlich angehaltener Container.

Ein weiteres wichtiges Feature ist die Fähigkeit, den Container im Hintergrund als einen Daemon-Prozess ausführen zu können. Die Option `-d` ermöglicht eine Ausführung im getrennten Modus. Bei Verwendung dieser Option gibt Podman die Container-ID auf dem Bildschirm zurück, sodass Sie die Befehle im selben Terminal ausführen können, während der Container im Hintergrund ausgeführt wird:

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

Das Container-Image gibt den auszuführenden Befehl an, um den containerisierten Prozess zu starten, der als Einstiegspunkt bezeichnet wird. Der Befehl `podman run` kann diesen Einstiegspunkt überschreiben, indem der Befehl nach dem Container-Image eingefügt wird:

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

Der angegebene Befehl muss innerhalb des Container-Images ausführbar sein.



Anmerkung

Da ein bestimmter Befehl im Beispiel angezeigt wird, überspringt der Container den Einstiegspunkt für das `httpd`-Image. Daher wird der `httpd`-Service nicht gestartet.

Kapitel 3 | Verwalten von Containern

Einige Container müssen als interaktive Shell oder als interaktiver Prozess ausgeführt werden. Dazu zählen Container, die Prozesse mit erforderlichen Benutzereingaben (wie das Eingeben von Befehlen) ausführen, und Prozesse, die durch Standardausgabe eine Ausgabe erzeugen. Das folgende Beispiel startet eine interaktive bash-Shell in einem `registry.redhat.io/rhel8/httpd-24`-Container:

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

Die Optionen `-t` und `-i` aktivieren die Terminalumleitung für interaktive textbasierte Programme. Die Option `-t` ordnet ein (Terminal) `pseudo-tty` zu und hängt es an die Standardeingabe des Containers an. Die Option `-i` hält die Standardeingabe des Containers offen, selbst wenn er getrennt wurde, sodass der Hauptprozess weiterhin auf die Eingabe warten kann.

Ausführen von Befehlen in einem Container

Wenn ein Container gestartet wird, führt er den Einstiegspunktbefehl aus. Manchmal kann es erforderlich sein, zur Verwaltung des ausgeführten Containers andere Befehle auszuführen.

Im Folgenden finden Sie einige typische Anwendungsfälle:

- Ausführen einer interaktiven Shell in einem bereits laufenden Container.
- Ausführen von Prozessen, welche die Dateien des Containers aktualisieren oder anzeigen.
- Starten neuer Hintergrundprozesse innerhalb des Containers.

Mit dem Befehl `podman exec` wird ein weiterer Prozess in einem bereits ausgeführten Container gestartet:

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

In diesem Beispiel wird die Container-ID verwendet, um einen Befehl in einem vorhandenen Container auszuführen.

Podman merkt sich den letzten Container, der in einem Befehl verwendet wird. Entwickler können das Schreiben der ID oder des Namens dieses Containers in späteren Podman-Befehlen überspringen, indem sie die Container-ID durch die Option `-l` ersetzen:

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

Verwalten von Containern

Das Erstellen und Starten eines Containers ist nur der erste Schritt im Lebenszyklus des Containers. Zu diesem Lebenszyklus gehören auch das Anhalten, Neustarten oder Entfernen des Containers. Benutzer können auch den Container-Status und die Metadaten auf Fehlerbehebung, Aktualisierung oder Berichterstellung prüfen.

Podman bietet die folgenden Befehle zum Verwalten von Containern:

- `podman ps`: Mit diesem Befehl werden die ausgeführten Container aufgelistet:

```
[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 In jeder Zeile werden Informationen zum Container beschrieben.

Jeder Container erhält nach der Erstellung eine **Container-ID**. Dies ist eine Hexadezimalzahl. Diese ID ähnelt einer Image-ID, ist jedoch nicht zugeordnet.

Im Feld **IMAGE** wird das Container-Image angegeben, das zum Starten des Containers verwendet wurde.

Im Feld **COMMAND** wird der Befehl angegeben, der beim Starten des Containers ausgeführt wurde.

Im Feld **CREATED** werden das Datum und die Uhrzeit des Starts des Containers angegeben.

Im Feld **STATUS** wird die Gesamtbetriebszeit des Containers (wenn dieser noch ausgeführt wird) oder die Zeitdauer nach dem Beenden angegeben.

Im Feld **PORTS** werden Ports angegeben, die über den Container oder eine Portweiterleitung bereitgestellt wurden, sofern konfiguriert.

Im Feld **NAMES** wird der Container-Namen angegeben.

Von Podman werden gestoppte Container nicht sofort verworfen. Podman behält ihre lokalen Dateisysteme und andere Statuswerte zum Nutzen der *postmortem*-Analyse bei. Option **-a** listet alle Container auf, einschließlich der angehaltenen:

```
[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-htt...
```



Anmerkung

Beim Erstellen von Containern bricht Podman ab, wenn der Containername bereits verwendet wird, selbst wenn sich der Container in einem Container befindet gestoppt Status. Diese Option hilft, doppelte Container-Namen zu vermeiden.

- **podman stop**: Mit diesem Befehl wird ein ausgeführter Container kontrolliert beendet:

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

Es ist einfacher, den Befehl **podman stop** zu verwenden, anstatt den Container zu suchen, den Prozess auf dem Hostbetriebssystem zu starten und anschließend zu beenden.

- **podman kill**: Dieser Befehl sendet Unix-Signale an den Hauptprozess im Container. Wenn kein Signal angegeben ist, wird das Signal **SIGKILL** gesendet, wodurch der Hauptprozess und der Container beendet werden.

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

Sie können das Signal mit der Option -s angeben:

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

Jedes Unix-Signal kann an den Hauptprozess gesendet werden. Podman akzeptiert entweder den Namen oder die Nummer des Signals.

In der folgenden Tabelle werden mehrere nützliche Signale erläutert:

Signal	Wert	Standardaktion	Kommentar
SIGHUP	1	Term	Abbruch im Kontrollterminal oder Ausfall des Kontrollprozesses erkannt
SIGINT	2	Term	Unterbrechung über die Tastatur
SIGQUIT	3	Core	Beendigung über die Tastatur
SIGILL	4	Core	Ungültige Anweisung
SIGABRT	6	Core	Abbruchsignal durch Abbruch (3)
SIGFPE	8	Core	Gleitkomma-Ausnahme
SIGKILL	9	Term	Kill-Signal
SIGSEGV	11	Core	Ungültiger Speicherverweis
SIGPIPE	13	Term	Beschädigtes Pipe-Zeichen: Schreiben in Pipe-Zeichen ohne Leser
SIGALRM	14	Term	Zeitgebersignal von Alarm(2)
SIGTERM	15	Term	Beendigungssignal
SIGUSR1	30,10,16	Term	Benutzerdefiniertes Signal 1
SIGUSR2	31,12,17	Term	Benutzerdefiniertes Signal 2
SIGCHLD	20,17,18	Ign	Untergeordnetes Element gestoppt oder beendet
SIGCONT	19,18,25	Cont	Bei angehaltenen Vorgang fortsetzen
SIGSTOP	17,19,23	Stop	Prozess stoppen
SIGTSTP	18,20,24	Stop	Stopp eingegeben bei tty
SIGTTIN	21,21,26	Stop	tty-Eingabe für den Hintergrundprozess
SIGTTOU	22,22,27	Stop	tty-Ausgabe für den Hintergrundprozess

Jedes Unix-Signal kann an den Hauptprozess gesendet werden. Podman akzeptiert entweder den Namen oder die Nummer des Signals.

**Anmerkung****Begriff**

Beenden Sie den Prozess.

Core

Beenden Sie den Prozess und erstellen Sie einen Speicherauszug.

Ign

Das Signal wird ignoriert.

Stop

Stoppen Sie den Prozess.

Zu den weiteren nützlichen Podman-Befehlen zählen:

- `podman restart`: Mit diesem Befehl wird ein angehaltener Container neu gestartet:

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

Mithilfe des Befehls `podman restart` wird ein neuer Container mit derselben Container-ID erstellt, wobei Status und Dateisystem des beendeten Containers wiederverwendet werden.

*Der Befehl `podman rm` löscht einen Container und verwirft dessen Status und Dateisystem.

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

Die Option `-f` des Unterbefehls `rm` weist Podman an, den Container zu entfernen, auch wenn er nicht gestoppt ist. Diese Option erzwingt die Beendigung des Containers und entfernt ihn dann. Die Verwendung der Option `-f` entspricht der gemeinsamen Verwendung der Befehle `podman kill` und `podman rm`.

Sie können sämtliche Container gleichzeitig löschen. Die Option `-a` wird von vielen `podman`-Unterbefehlen akzeptiert. Diese Option zeigt an, dass der Unterbefehl für alle verfügbaren Container oder Bilder verwendet wird. Im folgenden Beispiel werden alle Container entfernt:

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

Bevor Sie sämtliche Container löschen, müssen alle ausgeführten Container mit angehaltenen Status beendet werden. Mit dem folgenden Befehl können Sie alle Container stoppen:

+

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

Bevor Sie sämtliche Container löschen, müssen alle ausgeführten Container mit angehaltenen Status beendet werden. Mit dem folgenden Befehl können Sie alle Container stoppen.



Anmerkung

Die Unterbefehle `inspect`, `stop`, `kill`, `restart`, und `rm` können die Container-ID anstelle des Container-Namens verwenden.



Literaturhinweise

Manpage mit Unix Posix-Signalen

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

► Angeleitete Übung

Verwalten von MySQL-Containern

In dieser Übung erstellen und verwalten Sie einen MySQL®-Datenbank-Container.

Ergebnisse

Sie sollten in der Lage sein, einen MySQL-Datenbank-Container zu erstellen und zu verwalten.

Bevor Sie Beginnen

Stellen Sie sicher, dass für den Rechner `workstation` der Befehl `podman` verfügbar und ordnungsgemäß eingerichtet ist, indem Sie den folgenden Befehl in einem Terminalfenster ausführen:

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

Anweisungen

- ▶ 1. Laden Sie das MySQL-Datenbank-Container-Image herunter, und versuchen Sie, es zu starten. Der Container wird nicht gestartet, da mehrere Umgebungsvariablen für das Image bereitgestellt werden müssen.
 - 1.1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an. Lesen Sie die Anweisungen in *Anhang D, Erstellen eines Red Hat-Benutzerkontos*, wenn Sie sich bei Red Hat registrieren müssen.

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

- 1.2. Laden Sie das MySQL-Datenbank-Container-Image herunter, und versuchen Sie, es zu starten.

```
[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
```



Anmerkung

Beim Versuch, den Container als Daemon auszuführen (-d), wird die Fehlermeldung bezüglich der erforderlichen Variablen nicht angezeigt. Diese Fehlermeldung ist Teil der Container-Protokolle und kann über den folgenden Befehl angezeigt werden:

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

- 2. Erstellen Sie einen neuen Container mit dem Namen `mysql`, und geben Sie dann jede erforderliche Variable mithilfe des Parameters `-e` an.



Anmerkung

Stellen Sie sicher, dass Sie den neuen Container mit dem korrekten Namen starten.

```
[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
```

Der Befehl zeigt die Container-ID für den `mysql`-Container an. Im Folgenden finden Sie ein Beispiel der Ausgabe.

```
a49dba9ff17f2b5876001725b581fdd331c9ab8b9eda21cc2a2899c23f078509
```

- 3. Überprüfen Sie, ob der Container `mysql` ordnungsgemäß gestartet wurde. Führen Sie den folgenden Befehl aus:

```
[student@workstation ~]$ podman ps --format="{{.ID}} {{.Names}} {{.Status}}"  
a49dba9ff17f mysql Up About a minute ago
```

Der Befehl zeigt nur die ersten 12 Zeichen der Container-ID an, die im vorherigen Befehl angezeigt wurden.

- 4. Füllen Sie die Datenbank `items` anhand der Tabelle `Projects`:

- 4.1. Führen Sie den Befehl `podman cp` aus, um die Datenbankdatei in den Container `mysql` zu kopieren.

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

- 4.2. Füllen Sie die Datenbank `items` anhand der Tabelle `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. Erstellen Sie einen weiteren Container unter Verwendung desselben Container-Images aus dem vorherigen Container. Rufen Sie die Shell /bin/bash interaktiv auf, anstatt den Standardbefehl für das Container-Image zu verwenden.

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

- 6. Versuchen Sie, eine Verbindung zur MySQL-Datenbank im neuen Container herzustellen:

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

Der folgende Fehler wird angezeigt:

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

Der MySQL-Datenbankserver wird nicht ausgeführt, da der Container den Befehl /bin/bash ausgeführt hat, anstatt den MySQL-Server zu starten.

- 7. Beenden Sie den Container.

```
bash-4.4$ exit
```

- 8. Verifizieren Sie, dass der Container mysql-2 nicht ausgeführt wird.

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

- 9. Führen Sie eine Abfrage für den Container mysql aus, um alle Zeilen in der Tabelle Projects aufzulisten. Der Befehl weist die bash-Shell an, die Datenbank items mit einem mysql-Befehl abzufragen.

```
[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    DO180
```

Beenden

Führen Sie auf workstation das Skript lab_manage-lifecycle_finish aus, um diese Übung zu beenden.

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

Hiermit ist diese Übung beendet.

Anhängen von persistentem Storage

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Containerübergreifende Speicherung von Anwendungsdaten beim Neustarten von Containern unter Verwendung von persistentem Storage
- Konfiguration von Host-Verzeichnissen für eine Verwendung als Container-Volumes
- Bereitstellen eines Volumes im Container

Vorbereiten von persistenten Speicherorten

Der Container-Storage wird als *temporär* klassifiziert. Das bedeutet, dass dessen Inhalte nach der Entfernung des Containers nicht beibehalten werden. Es wird davon ausgegangen, dass containerisierte Anwendungen immer mit einem leeren Storage gestartet werden. Dadurch wird die Erstellung und Löschung von Containern relativ kostengünstig.

Im Verlauf dieses Kurses wurden Container-Images als *unveränderlich* und *geschichtet* bezeichnet. Das bedeutet, dass sie niemals geändert werden, jedoch aus Ebenen bestehen, welche die Inhalte von untergeordneten Ebenen hinzufügen oder überschreiben.

Bei einem ausgeführten Container wird eine neue Ebene über dessen Container-Basis-Image gelegt, und diese Ebene ist der *Container-Storage*. Diese Ebene ist zunächst der einzige verfügbare Storage mit Schreib-/Lesezugriff für den Container und wird zur Erstellung von Arbeitsdateien, temporären Dateien und Protokolldateien verwendet. Diese Dateien gelten als temporär. Anwendungen werden bei einem Verlust nicht beendet. Die Container-Storage-Ebene ist dem ausgeführten Container vorbehalten. Wenn ein anderer Container über dasselbe Basis-Image erstellt wird, erhält er eine andere Ebene mit Schreib-/Lesezugriff. Dadurch wird sichergestellt, dass die Ressourcen jedes Containers von anderen ähnlichen Containern isoliert werden.

Temporärer Container-Storage ist für Anwendungen, bei denen Daten nach dem Neustart beibehalten werden müssen (z. B. Datenbanken), *nicht* ausreichend. Zur Unterstützung solcher Anwendungen muss der Administrator einen Container mit persistentem Storage bereitstellen.

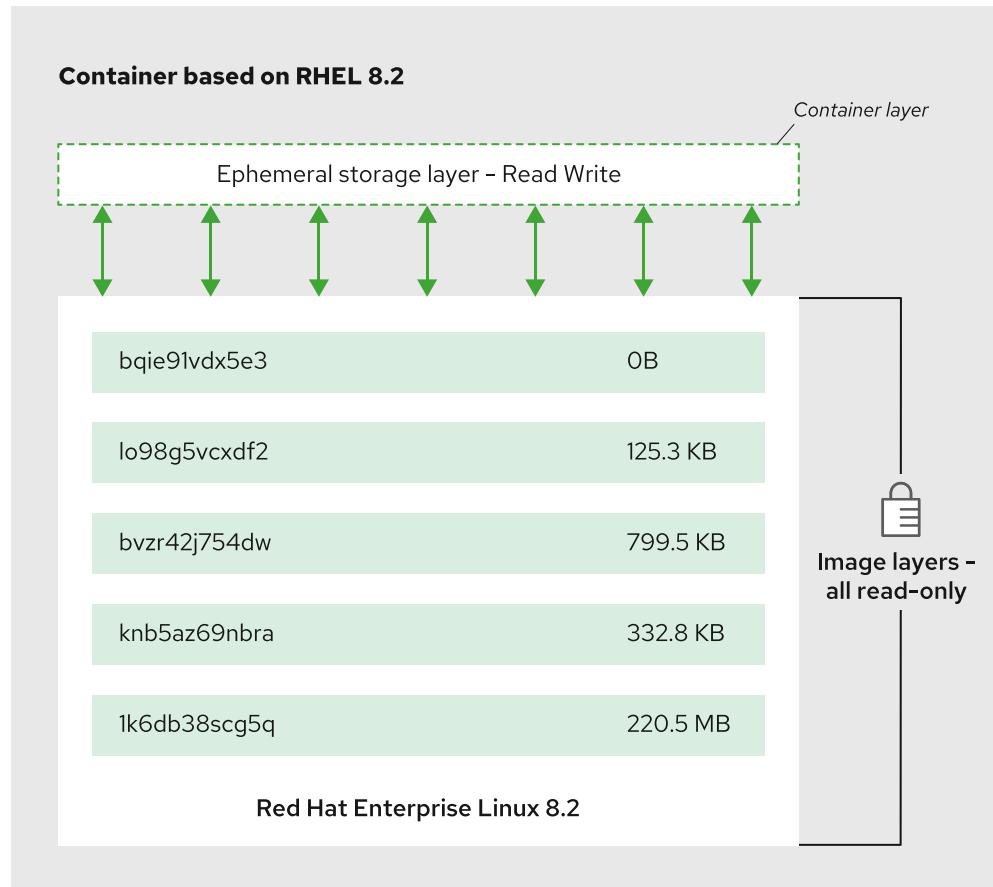


Abbildung 3.3: Container-Ebenen

Containerisierte Anwendungen sollten zur Speicherung persistenter Daten nicht den Container-Storage verwenden, da sie nicht steuern können, wie lange die entsprechenden Inhalte beibehalten werden.

Selbst wenn es möglich wäre, Container-Storage unendlich beizubehalten, eignet sich das geschichtete Dateisystem weder für intensive E-/A-Workloads noch für den Großteil der Anwendungen, die persistenten Storage erfordern.

Zurückfordern von Storage

Podman behält einen alten beendeten Container-Storage für Vorgänge zur Problembehebung bei, zum Beispiel zur Überprüfung fehlgeschlagener Container-Protokolle.

Wenn der Administrator alten Container-Storage zurückfordern muss, kann der Container mit `podman rm container_id` gelöscht werden. Dieser Befehl löscht auch den Container-Storage. Sie können die gestoppten Container-IDs mit dem Befehl `podman ps -a` suchen.

Vorbereiten des Hostverzeichnisses

Podman kann Hostverzeichnisse in einem laufenden Container mounten. Die containerisierte Anwendung erachtet diese Hostverzeichnisse als Teil des Container-Storages, so wie ein Remote-Netzwerkvolume von Anwendungen als Teil des Hostdateisystems angesehen wird. Die Inhalte dieses Hostverzeichnisses werden jedoch nach Beenden des Containers nicht zurückgefördert und können bei Bedarf für neue Container bereitgestellt werden.

Beispielsweise kann ein Datenbank-Container ein Hostverzeichnis verwenden, um Datenbankdateien zu speichern. Wenn dieser Datenbank-Container ausfällt, kann Podman einen neuen Container unter Verwendung desselben Hostverzeichnisses erstellen. Die Datenbankdaten werden dabei für Client-Anwendungen beibehalten. Aus Hostperspektive spielt es für den Datenbank-Container keine Rolle, wo dieses Verzeichnis gespeichert ist. Es kann sich überall befinden – auf einer lokalen Festplattenpartition bis hin zu einem externen Netzwerkdateisystem.

Ein Container wird unter Verwendung einer Benutzer- und Gruppen-ID des Hostbetriebssystems als Hostbetriebssystemprozess ausgeführt. Für das Hostverzeichnis müssen daher ein Eigentümer und Berechtigungen konfiguriert werden, die den Zugriff auf den Container ermöglichen. In RHEL muss das Hostverzeichnis zudem unter Verwendung des entsprechenden SELinux-Kontexts `container_file_t` konfiguriert werden. Podman verwendet den SELinux-Kontext `container_file_t`, um zu beschränken, auf welche Dateien des Hostsystems der Container zugreifen darf. Dadurch wird ein Datenleck zwischen dem Hostsystem und den Anwendungen, die in Containern ausgeführt werden, vermieden.

Nachfolgend wird eine Vorgehensweise zur Einrichtung des Hostverzeichnisses beschrieben:

1. Erstellen Sie ein Verzeichnis:

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

2. Der Benutzer, der Prozesse im Container ausführt, muss Dateien in das Verzeichnis schreiben können. Die Berechtigung sollte mit der numerischen Benutzer-ID (UID) aus dem Container definiert werden. Im Falle des von Red Hat bereitgestellten MySQL-Services lautet die UID 27. Der Befehl `podman unshare` bietet eine Sitzung zum Ausführen von Befehlen im selben Benutzer-Namespace wie der im Container ausgeführte Prozess.

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

3. Wenden Sie den `container_file_t`-Kontext auf das Verzeichnis (und alle Unterverzeichnisse) an, um Containern Zugriff auf alle Inhalte zu gewähren.

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

4. Wenden Sie die im ersten Schritt eingerichtete SELinux-Container-Richtlinie auf das neu erstellte Verzeichnis an:

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

Das Hostverzeichnis muss konfiguriert werden, bevor der Container gestartet wird, der das Verzeichnis verwendet.

Bereitstellen von Volumes

Nach der Erstellung und Konfiguration des Hostverzeichnisses muss dieses Verzeichnis in einem Container bereitgestellt werden. Um ein Hostverzeichnis in einem Container bereitzustellen, fügen Sie die Option `-v` zum Befehl `podman run` hinzu; geben Sie dabei den Pfad des Hostverzeichnisses und den Pfad des Container-Storage an, durch einen Doppelpunkt (`:`) getrennt.

Beispiel: Verwenden Sie den folgenden Befehl, um das Hostverzeichnis `/home/student/dbfiles` für Datenbankdateien des MySQL-Servers zu verwenden, die sich erwartungsgemäß unter `/var/lib/mysql` in einem MySQL-Container-Image namens `mysql` befinden:

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

Wenn in diesem Befehl `/var/lib/mysql` bereits im Container-Image `mysql` vorhanden ist, wird die Bereitstellung `/home/student/dbfiles` überlagert; die Inhalte aus dem Container-Image werden jedoch nicht entfernt. Wenn die Bereitstellung entfernt wird, sind die ursprünglichen Inhalte wieder zugänglich.

► Angeleitete Übung

MySQL-Container mit persistenter Datenbank erstellen

In dieser Übung erstellen Sie einen Container, der die MySQL-Datenbankdaten in einem Hostverzeichnis speichert.

Ergebnisse

Sie sollten in der Lage sein, einen Container mit einer persistenten Datenbank bereitzustellen.

Bevor Sie Beginnen

Auf workstation sollten keine Container-Images ausgeführt werden.

Führen Sie den folgenden Befehl auf workstation:

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

Anweisungen

- 1. Erstellen Sie das Verzeichnis /home/student/local/mysql mit dem richtigen SELinux-Kontext und den richtigen Berechtigungen.
- 1.1. Erstellen Sie das Verzeichnis /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. Fügen Sie den entsprechenden SELinux-Kontext für das Verzeichnis /home/student/local/mysql und dessen Inhalte hinzu.

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

- 1.3. Wenden Sie die SELinux-Richtlinie auf das neu erstellte Verzeichnis an.

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

- 1.4. Verifizieren Sie, dass container_file_t der SELinux-Kontexttyp für das Verzeichnis /home/student/local/mysql ist.

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

- 1.5. Ändern Sie den Eigentümer des Verzeichnisses /home/student/local/mysql in den Benutzer mysql und die Gruppe mysql:

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



Anmerkung

Der Benutzer, der Prozesse im Container ausführt, muss Dateien in das Verzeichnis schreiben können.

Die Berechtigung sollte mit der numerischen Benutzer-ID (UID) aus dem Container definiert werden. Im Falle des von Red Hat bereitgestellten MySQL-Services lautet die UID 27. Der Befehl podman unshare bietet eine Sitzung zum Ausführen von Befehlen im selben Benutzer-Namespace wie der im Container ausgeführte Prozess.

- 2. Erstellen Sie eine MySQL-Containerinstanz mit persistentem Storage:

- 2.1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an. Lesen Sie die Anweisungen in *Anhang D, Erstellen eines Red Hat-Benutzerkontos*, wenn Sie sich bei Red Hat registrieren müssen.

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

- 2.2. Rufen Sie das MySQL-Container-Image ab.

```
[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. Erstellen Sie einen neuen Container, und geben Sie den Mount-Punkt zur Speicherung der MySQL-Datenbankdaten an:

```
[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
```

Mit diesem Befehl wird das Verzeichnis /home/student/local/mysql vom Host im Verzeichnis /var/lib/mysql/data im Container bereitgestellt. Standardmäßig speichert die MySQL-Datenbank Daten im Verzeichnis /var/lib/mysql/data.

- 2.4. Überprüfen Sie, ob der Container ordnungsgemäß gestartet wurde.

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

- 3. Überprüfen Sie, ob das Verzeichnis /home/student/local/mysql das Verzeichnis items enthält:

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

Im Verzeichnis items werden die für die items-Datenbank relevanten Daten, die von diesem Container erstellt wurde, gespeichert. Wenn das Verzeichnis items nicht verfügbar ist, wurde der Mount-Punkt während der Container-Erstellung nicht korrekt definiert.



Anmerkung

Alternativ können Sie denselben Befehl mit podman unshare ausführen, um die numerische Benutzer-ID (UID) über den Container zu überprüfen.

```
[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
```

Beenden

Führen Sie auf workstation das Skript lab manage-storage finish aus, um diese praktische Übung abzuschließen.

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

Hiermit ist diese Übung beendet.

Zugreifen auf Container

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Beschreiben der Grundlagen zur Vernetzung mit Containern
- Herstellen von Remote-Verbindungen zu Services innerhalb eines Containers

Zuordnen von Netzwerk-Ports

Der Zugriff auf einen rootless-Container über das Hostnetzwerk kann schwierig sein, da für einen rootless-Container keine IP-Adresse verfügbar ist.

Definieren Sie zum Lösen dieser Probleme Portweiterleitungsregeln, um den externen Zugriff auf einen Container-Service zu ermöglichen. Verwenden Sie die Option `-p [<IP address>:] [<host port>:]<container port>` mit dem Befehl `podman run`, um einen extern zugänglichen Container zu erstellen.

Sehen Sie sich folgendes Beispiel an:

```
[user@host ~]$ podman run -d --name apache1 -p 8080:80 \
> registry.redhat.io/rhel8/httpd-24
```

In diesem Beispiel wird ein extern zugänglicher Container erstellt. Der Wert 8080:80 gibt an, dass alle Anforderungen an Port 8080 auf dem Host an Port 80 innerhalb des Containers weitergeleitet werden.

Sie können auch die Option `-p` verwenden, um den Port an die angegebene IP-Adresse zu binden.

```
[user@host ~]$ podman run -d --name apache2 \
> -p 127.0.0.1:8081:80 registry.redhat.io/rhel8/httpd-24
```

Dieses Beispiel beschränkt den externen Zugriff auf den apache2-Container auf Anfragen von localhost zum Hostport 8081. Diese Anforderungen werden an den Port 80 im apache2-Container weitergeleitet.

Wenn für den Hostport kein Port angegeben wurde, weist Podman einen zufällig verfügbaren Hostport für den Container zu:

```
[user@host ~]$ podman run -d --name apache3 -p 127.0.0.1::80 \
> registry.redhat.io/rhel8/httpd-24
```

Führen Sie den Befehl `podman port <container name>` aus, um den von Podman zugewiesenen Port anzuzeigen:

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

Wenn nur ein Container-Port mit der Option `-p` angegeben ist, wird dem Container ein zufällig verfügbarer Hostport zugewiesen. Anforderungen an diesen zugewiesenen Hostport von einer beliebigen IP-Adresse werden an den Container-Port weitergeleitet.

```
[user@host ~]$ podman run -d --name apache4 \  
> -p 80 registry.redhat.io/rhel8/httpd-24  
[user@host ~]$ podman port apache4  
80/tcp -> 0.0.0.0:37068
```

In diesem Beispiel wird jede Routing-fähige Anforderung an Hostport 37068 an Port 80 im Container weitergeleitet.



Literaturhinweise

Container-Netzwerkschnittstelle – Networking für Linux-Container

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

Cloud Native Computing Foundation

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

► Angeleitete Übung

Laden der Datenbank

In dieser Übung erstellen Sie einen MySQL-Datenbank-Container mit aktivierter Port-Weiterleitung. Nachdem Sie eine Datenbank mit einem SQL-Skript gefüllt haben, verifizieren Sie den Datenbankinhalt mit drei unterschiedlichen Methoden.

Ergebnisse

Sie sollten in der Lage sein, einen Datenbank-Container bereitzustellen und ein SQL-Skript zu laden.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf dem Rechner `workstation` einen Terminal, und führen Sie den folgenden Befehl aus:

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

Dadurch wird gewährleistet, dass das Verzeichnis `/home/student/local/mysql` vorhanden und mit den richtigen Berechtigungen konfiguriert ist, um den permanenten Storage für den MySQL-Container zu aktivieren.

Anweisungen

- ▶ 1. Erstellen Sie eine MySQL-Container-Instanz mit persistentem Storage und Portweiterleitung.
 - 1.1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an. Lesen Sie die Anweisungen in *Anhang D, Erstellen eines Red Hat-Benutzerkontos*, wenn Sie sich bei Red Hat registrieren müssen.

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

- 1.2. Laden Sie das MySQL-Datenbank-Container-Image herunter, und erstellen Sie dann eine MySQL-Container-Instanz mit persistentem Storage und Portweiterleitung.

```
[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
```

Die letzte Zeile Ihrer Ausgabe und die Zeit, die zum Herunterladen der einzelnen Image-Ebenen benötigt wird, unterscheiden sich.

Mit der Option -p wird die Port-Weiterleitung so konfiguriert, dass Port 13306 auf dem lokalen Host an den Container-Port 3306 weitergeleitet wird.



Anmerkung

Das Startskript erstellt das Verzeichnis /home/student/local/mysql mit dem entsprechenden Eigentümer- und SELinux-Kontext, der für die Container-Datenbank erforderlich ist.

- ▶ 2. Verifizieren Sie, dass der Container mysqlDb-port erfolgreich gestartet wurde, und aktivieren Sie die Port-Weiterleitung.

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

- ▶ 3. Füllen Sie die Datenbank mit der bereitgestellten Datei. Wenn keine Fehler vorliegen, gibt der Befehl keine Ausgabe zurück.

```
[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.
```

Es gibt mehrere Möglichkeiten, um zu verifizieren, dass die Datenbank erfolgreich geladen wurde. Die nächsten Schritte zeigen drei verschiedene Methoden. Sie müssen nur eine der Methoden verwenden.

- ▶ 4. Verifizieren Sie, dass die Datenbank erfolgreich geladen wurde, indem Sie einen nicht interaktiven Befehl im Container ausführen.

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

Kapitel 3 | Verwalten von Containern

- 5. Verifizieren Sie, dass die Datenbank erfolgreich geladen wurde, indem Sie die Port-Weiterleitung vom lokalen Host verwenden. Diese alternative Methode ist optional.

```
[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. Verifizieren Sie, dass die Datenbank erfolgreich geladen wurde, indem Sie eine interaktive Terminalsitzung im Container öffnen. Diese alternative Methode ist optional.

- 6.1. Öffnen Sie eine Bash-Shell im Container.

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

- 6.2. Verifizieren Sie, dass die Datenbank Daten enthält:

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

- 6.3. Beenden Sie den Container:

```
bash-4.4$ exit
```

Beenden

Führen Sie auf workstation das Skript lab manage-networking finish aus, um diese praktische Übung abzuschließen.

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

Hiermit ist diese Übung beendet.

► Praktische Übung

Verwalten von Containern

Ergebnisse

Sie sollten in der Lage sein, eine persistente Datenbank mithilfe eines freigegebenen Volumes bereitzustellen und zu verwalten. Zudem sollten Sie in der Lage sein, eine zweite Datenbank mit demselben freigegebenen Volume zu starten und zu überwachen, ob die Daten zwischen den zwei Containern konsistent sind, da sie dasselbe Verzeichnis auf dem Host verwenden, um die MySQL-Daten zu speichern.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Erstellen Sie das Verzeichnis `/home/student/local/mysql` mit dem richtigen SELinux-Kontext und den richtigen Berechtigungen.
2. Stellen Sie eine MySQL-Container-Instanz unter Verwendung der folgenden Merkmale bereit:
 - *Name*: `mysql-1`
 - *Als Daemon ausführen*: ja
 - *Volume*: vom Hostordner `/home/student/local/mysql` zum Container-Ordner `/var/lib/mysql/data`
 - *Container-Image*: `registry.redhat.io/rhel8/mysql-80:1`
 - *Portweiterleitung*: ja, von Host-Port 13306 zu Container-Port 3306
 - *Umgebungsvariablen*:
 - `MYSQL_USER`: `user1`
 - `MYSQL_PASSWORD`: `mypa55`
 - `MYSQL_DATABASE`: `items`
 - `MYSQL_ROOT_PASSWORD`: `r00tpa55`
3. Laden Sie die Datenbank `items` mit dem Skript `/home/student/D0180/labs/manage-review/db.sql`.
4. Beenden Sie kontrolliert den Container.



Wichtig

Dieser Schritt ist sehr wichtig, da ein neuer Container erstellt wird, der für Datenbankdaten dasselbe Volume verwendet. Wenn zwei Container dasselbe Volume verwenden, kann die Datenbank beschädigt werden. Führen Sie keinen Neustart des mysql-1-Containers durch.

5. Erstellen Sie einen neuen Container mit den folgenden Merkmalen:
 - *Name:* mysql-2
 - *Als Daemon ausführen:* ja
 - *Volume:* vom Hostordner /home/student/local/mysql zum Container-Ordner /var/lib/mysql/data
 - *Container-Image:* registry.redhat.io/rhel8/mysql-80:1
 - *Portweiterleitung:* ja, von Host-Port 13306 zu Container-Port 3306
 - *Umgebungsvariablen:*
 - MYSQL_USER: user1
 - MYSQL_PASSWORD: mypa55
 - MYSQL_DATABASE: items
 - MYSQL_ROOT_PASSWORD: r00tpa55
6. Speichern Sie die Liste aller Container (einschließlich der beendeten Container) in der Datei /tmp/my-containers.
7. Greifen Sie auf die Bash-Shell im Container zu und verifizieren Sie, dass die `items`-Datenbank und `Item`-Tabelle weiterhin verfügbar sind. Bestätigen Sie außerdem, dass die Tabelle Daten enthält.
8. Fügen Sie mittels Portweiterleitung eine neue Zeile in die Tabelle `Item` ein. Die Zeile sollte den `description`-Wert `Finished lab` und den `done`-Wert `1` aufweisen.
9. Da der erste Container nicht mehr benötigt wird, entfernen Sie ihn, um Ressourcen freizugeben.

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

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem `workstation`-Rechner den Befehl `lab manage-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf `workstation` den Befehl `lab manage-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Verwalten von Containern

Ergebnisse

Sie sollten in der Lage sein, eine persistente Datenbank mithilfe eines freigegebenen Volumes bereitzustellen und zu verwalten. Zudem sollten Sie in der Lage sein, eine zweite Datenbank mit demselben freigegebenen Volume zu starten und zu überwachen, ob die Daten zwischen den zwei Containern konsistent sind, da sie dasselbe Verzeichnis auf dem Host verwenden, um die MySQL-Daten zu speichern.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Erstellen Sie das Verzeichnis `/home/student/local/mysql` mit dem richtigen SELinux-Kontext und den richtigen Berechtigungen.
 - 1.1. Erstellen Sie das Verzeichnis `/home/student/local/mysql`.

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

- 1.2. Fügen Sie den entsprechenden SELinux-Kontext für das Verzeichnis `/home/student/local/mysql` und dessen Inhalte hinzu. Mit aktuellem Kontext können Sie dieses Verzeichnis in einem ausgeführten Container mounten.

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

- 1.3. Wenden Sie die SELinux-Richtlinie auf das neu erstellte Verzeichnis an.

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

- 1.4. Ändern Sie den Eigentümer des Verzeichnisses `/home/student/local/mysql` für das Container-Image `registry.redhat.io/rhel8/mysql-80:1` in den Benutzer `mysql` und die Gruppe `mysql`:

```
[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. Stellen Sie eine MySQL-Container-Instanz unter Verwendung der folgenden Merkmale bereit:

Kapitel 3 | Verwalten von Containern

- *Name:* mysql-1
- *Als Daemon ausführen:* ja
- *Volume:* vom Hostordner /home/student/local/mysql zum Container-Ordner /var/lib/mysql/data
- *Container-Image:* registry.redhat.io/rhel8/mysql-80:1
- *Portweiterleitung:* ja, von Host-Port 13306 zu Container-Port 3306
- *Umgebungsvariablen:*
 - MYSQL_USER: user1
 - MYSQL_PASSWORD: mypa55
 - MYSQL_DATABASE: items
 - MYSQL_ROOT_PASSWORD: r00tpa55

2.1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an.

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

2.2. Erstellen Sie den Container und starten Sie ihn.

```
[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
616azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cdd
```

2.3. Überprüfen Sie, ob der Container ordnungsgemäß gestartet wurde.

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

3. Laden Sie die Datenbank items mit dem Skript /home/student/D0180/labs/manage-review/db.sql.

3.1. Laden Sie die Datenbank mit den SQL-Befehlen in /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. Verwenden Sie zum Ausgeben sämtlicher Zeilen der Item-Tabelle eine SQL SELECT-Anweisung, um zu verifizieren, dass die Items-Datenbank geladen ist.



Anmerkung

Sie können dem Befehl `mysql` den Parameter `-e SQL` hinzufügen, um eine SQL-Anweisung auszuführen.

```
[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. Beenden Sie kontrolliert den Container.



Wichtig

Dieser Schritt ist sehr wichtig, da ein neuer Container erstellt wird, der für Datenbankdaten dasselbe Volume verwendet. Wenn zwei Container dasselbe Volume verwenden, kann die Datenbank beschädigt werden. Führen Sie keinen Neustart des `mysql-1`-Containers durch.

- 4.1. Beenden Sie den Container mithilfe des Befehls `podman`.

```
[student@workstation ~]$ podman stop mysql-1
616azfaa55x866e7eb18b1fd0423a5461d8f24c147b1ad8668b76e6167587cd
```

5. Erstellen Sie einen neuen Container mit den folgenden Merkmalen:

- *Name*: `mysql-2`
- *Als Daemon ausführen*: ja
- *Volume*: vom Hostordner `/home/student/local/mysql` zum Container-Ordner `/var/lib/mysql/data`
- *Container-Image*: `registry.redhat.io/rhel8/mysql-80:1`
- *Portweiterleitung*: ja, von Host-Port 13306 zu Container-Port 3306
- *Umgebungsvariablen*:
 - `MYSQL_USER`: `user1`

Kapitel 3 | Verwalten von Containern

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

5.1. Erstellen Sie den Container und starten Sie ihn.

```
[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=mpa55 \
> -e MYSQL_DATABASE=items -e MYSQL_ROOT_PASSWORD=r00tpa55 \
> registry.redhat.io/rhel8/mysql-80:1
281c0e2790e54cd5a0b8e2a8cb6e3969981b85cde8ac611bf7ea98ff78bdffbb
```

5.2. Überprüfen Sie, ob der Container ordnungsgemäß gestartet wurde.

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

6. Speichern Sie die Liste aller Container (einschließlich der beendeten Container) in der Datei /tmp/my-containers.

6.1. Verwenden Sie den Befehl podman, um die Informationen in /tmp/my-containers zu speichern.

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

7. Greifen Sie auf die Bash-Shell im Container zu und verifizieren Sie, dass die items-Datenbank und Item-Tabelle weiterhin verfügbar sind. Bestätigen Sie außerdem, dass die Tabelle Daten enthält.

7.1. Greifen Sie im Container auf die Bash-Shell zu.

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

7.2. Stellen Sie eine Verbindung zum MySQL-Server her.

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

7.3. Listen Sie alle Datenbanken auf und überprüfen Sie, ob die items-Datenbank verfügbar ist.

```
mysql> show databases;
-----
| Database      |
-----
| information_schema |
| items          |
| mysql          |
| performance_schema |
```

```
| sys           |
-----
5 rows in set (0.03 sec)
```

- 7.4. Listen Sie alle Tabellen aus der `items`-Datenbank auf und überprüfen Sie, ob die `Item`-Tabelle verfügbar ist.

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

- 7.5. Zeigen Sie die Daten aus der Tabelle an.

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

- 7.6. Beenden Sie den MySQL-Client und die Container-Shell.

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

8. Fügen Sie mittels Portweiterleitung eine neue Zeile in die Tabelle `Item` ein. Die Zeile sollte den `description`-Wert `Finished lab` und den `done`-Wert `1` aufweisen.

- 8.1. Stellen Sie eine Verbindung zur MySQL-Datenbank her.

```
[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. Fügen Sie die neue Zeile ein.

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

8.3. Beenden Sie den MySQL-Client.

```
mysql> exit  
Bye
```

9. Da der erste Container nicht mehr benötigt wird, entfernen Sie ihn, um Ressourcen freizugeben.

Führen Sie den folgenden Befehl aus, um den Container zu entfernen:

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

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem workstation-Rechner den Befehl `lab manage-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf workstation den Befehl `lab manage-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Podman besitzt Unterbefehle zum Erstellen eines neuen Containers (`run`), zum Löschen eines Containers (`rm`), zum Auflisten von Containern (`ps`), zum Stoppen eines Containers (`stop`) und zum Starten eines Prozesses in einem Container (`exec`).
- Der standardmäßige Container-Storage ist temporär. Seine Inhalte sind nach dem Neustarten oder Entfernen des Containers folglich nicht mehr vorhanden.
- Container können einen Ordner aus dem Hostdateisystem verwenden, um persistente Daten zu verwenden.
- Podman stellt Volumes in einem Container mit der Option `-v` im Befehl `podman run` bereit.
- Mit dem Befehl `podman exec` wird ein weiterer Prozess in einem ausgeführten Container gestartet.
- Mithilfe der Option `-p` im Unterbefehl `run` ordnet Podman Container-Ports lokale Ports zu.

Kapitel 4

Verwalten von Container-Images

Ziel

Verwalten des Lebenszyklus eines Container-Images von der Erstellung bis zur Löschung

Ziele

- Suchen und Abrufen von Images von Remote-Registries
- Exportieren, Importieren und Verwalten von Container-Images lokal und in einer Registry

Abschnitte

- Zugreifen auf Registries (und Test)
- Ändern von Container-Images (und angeleitete Übung)

Praktische Übung

- Verwalten von Container-Images

Zugreifen auf Registries

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Suchen mithilfe von Podman-Befehlen und der REST-API der Registry nach Images in Remote-Registries und abrufen dieser
- Aufführen der sich aus der Verwendung einer zertifizierten öffentlichen Registry zum Herunterladen von sicheren Images ergebenden Vorteile
- Anpassen der Podman-Konfiguration zum Zugreifen auf alternative Container-Image-Registries
- Aufführen der von einer Registry in das lokale Dateisystem heruntergeladenen Images
- Verwalten von Tags zum Abrufen von mit Tags versehenen Images

Öffentliche Registries

Image-Registries sind Services, die herunterladbare Container-Images anbieten. Sie ermöglichen es Image-Erstellern und -Verwaltern, Container-Images zu speichern und an öffentliche oder private Zielgruppen zu verteilen.

Podman sucht nach Container-Images aus öffentlichen und privaten Registries und lädt die Container-Images herunter. Red Hat Container Catalog ist die öffentliche Image-Registry, die von Red Hat verwaltet wird. Darin wird ein großer Satz an Container-Images gehostet. Dazu zählen die von großen Open-Source-Projekten, beispielsweise Apache, MySQL und Jenkins, bereitgestellten. Alle Images im Container Catalog werden vom internen Red Hat-Sicherheitsteam untersucht, das heißt, sie sind vertrauenswürdig und gegen Sicherheitslücken geschützt.

Red Hat-Container-Images bieten die folgenden Vorteile:

- *Vertrauenswürdige Quelle*: Alle Container-Images beinhalten Quellen, die Red Hat bekannt und vertrauenswürdig sind.
- *Ursprüngliche Abhängigkeiten*: Die Container-Pakete wurden nicht manipuliert und enthalten ausschließlich bekannte Bibliotheken.
- *Frei von Sicherheitslücken*: Container-Images enthalten keine bekannten Sicherheitslücken in den Plattformkomponenten oder -ebenen.
- *Laufzeitschutz*: Alle Anwendungen in Container-Images werden als Nicht-Root-Benutzer ausgeführt, wodurch die Angriffsfläche für bösartige oder fehlerhafte Anwendungen minimiert wird.
- *Mit Red Hat Enterprise Linux (RHEL) kompatibel*: Container-Images sind mit allen RHEL-Plattformen kompatibel – von Bare Metal bis zur Cloud.
- *Red Hat-Unterstützung*: Der vollständige Stack wird kommerziell durch Red Hat unterstützt.

Quay.io ist ein weiteres öffentliches Image-Repository, das von Red Hat gesponsert wird. Quay.io bietet einige sehr nützliche Features. Dazu zählen beispielsweise die serverseitige Image-

Erstellung, detaillierte Zugriffskontrollen und die automatische Überprüfung von Images auf bekannte Schwachstellen.

Die Images von Red Hat Container Catalog sind vertrauenswürdig und verifiziert, Quay.io bietet aber Live-Images, die von Entwicklern regelmäßig aktualisiert werden. Quay.io-Benutzer können Ihre Namespaces mit detaillierter Zugriffskontrolle erstellen und die Images, die von ihnen erstellt werden, in diesem Namespace veröffentlichen. Container Catalog-Benutzer übertragen selten oder niemals neue Images. Vielmehr verwenden sie die vom Red Hat-Team generierten vertrauenswürdigen Images.

Private Registries

Image-Ersteller oder -Verwalter möchten ihre Images öffentlich zur Verfügung stellen. Andere Image-Ersteller bevorzugen es jedoch, ihre Images aus folgenden Gründen geheim zu halten:

- Privatsphäre und Geschäftsgeheimnis des Unternehmens.
- Gesetzliche Beschränkungen und Gesetze.
- Vermeidung der Veröffentlichung von in der Entwicklung befindlichen Images.

In einigen Fällen werden private Images bevorzugt. Private Registries geben Image-Erstellern die Kontrolle über die Platzierung, Verteilung und Verwendung ihrer Images.

Konfigurieren von Registries in Podman

Zum Konfigurieren von Registries für den Befehl `podman` müssen Sie die Datei `/etc/containers/registries.conf` aktualisieren. Bearbeiten Sie den Eintrag `registries` im Abschnitt `[registries.search]`, wodurch der Wertelist ein Eintrag hinzugefügt wird.

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



Anmerkung

Verwenden Sie einen FQDN und eine Portnummer, um eine Registry zu identifizieren. Eine Registry, die keine Portnummer enthält, hat eine Standardportnummer von 5000. Wenn die Registry einen anderen Port verwendet, muss dieser angegeben werden. Geben Sie die Portnummern an, indem Sie einen Doppelpunkt (:) und die Portnummer nach dem FQDN anhängen.

Für sichere Verbindungen zu einer Registry ist ein vertrauenswürdiges Zertifikat erforderlich. Fügen Sie dem Eintrag `registries` im Abschnitt `[registries.insecure]` der Datei `/etc/containers/registries.conf` den Registry-Namen hinzu, um unsichere Verbindungen zu unterstützen:

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

Zugreifen auf Registries

Podman bietet Befehle, mit denen Sie nach Images suchen können. Zudem lassen sich Images-Registries über eine API aufrufen. Im Folgenden werden beide Ansätze erläutert.

Suchen nach Images in öffentlichen Registries

Mit dem Befehl `podman search` wird in allen in der Konfigurationsdatei `/etc/containers/registries.conf` aufgelisteten Registries nach dem Image-Namen, Benutzernamen oder nach der Beschreibung gesucht. Die Syntax für den Befehl `podman search` findet sich im Folgenden:

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

Die folgende Tabelle enthält einige nützliche Optionen, die für den Unterbefehl `search` zur Verfügung stehen:

Option	Beschreibung
<code>--limit <number></code>	Begrenzt die Anzahl der aufgelisteten Images pro Registry.
<code>--filter <filter=value></code>	Filtern Sie die Ausgabe basierend auf den angegebenen Bedingungen. Die folgenden Filter werden unterstützt: stars=<number>: Nur Images mit mindestens dieser Anzahl an Sternen anzeigen. is-automated=<true false>: Nur automatisch erstellte Images anzeigen. is-official=<true false>: Nur Images anzeigen, die als offiziell gekennzeichnet sind.
<code>--tls-verify <true false></code>	Bestimmt, ob die Verbindung über TLS verschlüsselt ist. Der Standardwert ist <code>false</code> .

Registry-HTTP-API

Eine Remote-Registry stellt Webservices bereit, die der Registry eine Programmierschnittstelle (Application Programming Interface, API) bereitstellen. Podman verwendet diese Schnittstellen, um auf entfernte Repositorys zuzugreifen und mit ihnen zu interagieren. Viele Registrys entsprechen der Docker Registry HTTP API v2-Spezifikation, die eine standardisierte REST-Schnittstelle für die Registry-Interaktion bereitstellt. Sie können diese REST-Schnittstelle verwenden, um direkt mit einer Registry zu interagieren, anstatt Podman zu verwenden.

Beispiele für die Verwendung dieser API mit `curl`-Befehlen werden unten gezeigt:

Verwenden Sie den Endpunkt `/v2/_catalog`, um alle in einer Registry verfügbaren Repositorys aufzulisten. Der Parameter `n` wird verwendet, um die Anzahl der zurückzugebenden Repositorys zu begrenzen.

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

Verwenden Sie den Endpunkt `/v2/_catalog`, um alle in einer Registry verfügbaren Repositorys aufzulisten. Der Parameter `n` wird verwendet, um die Anzahl der zurückzugebenden Repositorys zu begrenzen.



Anmerkung

Wenn Python verfügbar ist, verwenden Sie es, um die JSON-Antwort zu formatieren:

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

Der Endpunkt /v2/<name>/tags/list enthält die Liste der für ein einzelnes Image verfügbaren Tags:

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



Anmerkung

Quay.io bietet eine dedizierte API für die Interaktion mit Repositorys über das hinaus, was in der Docker-Repository-API angegeben ist. Einzelheiten finden Sie in <https://docs.quay.io/api/>.

Registry-Authentifizierung

Einige Container-Image-Registries benötigen eine Zugriffsberechtigung. Der Befehl `podman login` erlaubt die Authentifizierung von Benutzername und Passwort in einer Registry:

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

Die Registry-HTTP-API erfordert Authentifizierungsinformationen. Verwenden Sie zuerst den Red Hat Single Sign On-Service (SSO), um ein Zugriffstoken abzurufen:

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

Fügen Sie dieses Token dann in nachfolgenden Anforderungen in ein Bearer-Autorisierungstoken ein:

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



Anmerkung

Bei anderen Registries sind möglicherweise andere Schritte erforderlich, um die Anmelddaten bereitzustellen. Wenn sich eine Registry an die Docker Registry HTTP v2 API hält, entspricht die Authentifizierung dem RFC7235-Schema.

Abrufen von Images

Führen Sie den Befehl `podman pull` aus, um Container-Images aus einer Registry abzurufen:

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

Der Befehl `podman pull` verwendet den Image-Namen, der vom Unterbefehl `search` abgerufen wurde, um ein Image aus einer Registry abzurufen. Mit dem Unterbefehl `pull` kann dem Image der Registry-Name hinzugefügt werden. Diese Variante ermöglicht, dass dasselbe Image in mehreren Registries vorhanden sein kann.

Verwenden Sie den folgenden Befehl, um beispielsweise einen NGINX-Container aus der Registry `quay.io` abzurufen:

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



Anmerkung

Wenn der Image-Name keinen Registry-Namen enthält, sucht Podman nach einem übereinstimmenden Container-Image unter Verwendung der in der Konfigurationsdatei `/etc/containers/registries.conf` aufgelisteten Registries. Podman sucht nach Images in Registries in derselben Reihenfolge, in der sie in der Konfigurationsdatei angezeigt werden.

Aufführen lokaler Image-Kopien

Jedes aus einer Registry heruntergeladene Container-Image wird lokal auf demselben Host gespeichert, auf dem der Befehl `podman` ausgeführt wird. Dieses Verhalten vermeidet das erneute Herunterladen von Images und minimiert die Bereitstellungszeit für einen Container. Podman speichert auch alle benutzerdefinierten Container-Images, die Sie im selben lokalen Storage erstellen.



Anmerkung

Standardmäßig speichert Podman Container-Images im Verzeichnis `/var/lib/containers/storage/overlay-images`.

Podman bietet den Unterbefehl `images`, um alle lokal gespeicherten Container-Images aufzulisten:

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

Image-Tags

Ein Image-Tag ist ein Mechanismus zum Unterstützen mehrerer Versionen desselben Images. Dieses Feature ist nützlich, wenn mehrere Versionen derselben Software, beispielsweise ein produktionsbereiter Container, oder die neuesten Updates derselben für die Community-Bewertung entwickelten Software bereitgestellt werden. Jeder Podman-Sub-Befehl, der einen Container-Image-Namen erfordert, akzeptiert einen Tag-Parameter, um zwischen mehreren Tags zu unterscheiden. Wenn ein Image-Name kein Tag enthält, wird standardmäßig der Tag-Wert `latest` verwendet. Verwenden Sie den folgenden Befehl, wenn Sie beispielsweise ein Image mit dem Tag `1` aus `rhel8/mysql-80` abrufen möchten:

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

Verwenden Sie den folgenden Befehl, um einen neuen Container auf Basis des Images `rhel8/mysql-80:1` zu starten:

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



Literaturhinweise

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: Authentifizierung

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

► Quiz

Verwenden von Registries

Wählen Sie anhand der folgenden Informationen die richtigen Antworten auf die folgenden Fragen aus:

Podman ist auf einem RHEL-Host mit dem folgenden Eintrag in der Datei /etc/containers/registries.conf verfügbar:

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

Die Hosts `registry.redhat.io` und `quay.io` verfügen über eine aktive Registry, weisen beide gültige Zertifikate auf und verwenden Version 1 der Registry. Die folgenden Images sind für jeden Host verfügbar:

Image-Namen/Tags pro Registry

Registry	Image
registry.redhat.io	rhel8/nginx/1.1.6
	rhel8/mysql-8.0
	rhel8/httpd-2.4
quay.io	mysql-5.7
	httpd-2.4

Es sind keine Images lokal verfügbar.

- ▶ 1. Welche zwei Befehle zeigen die verfügbaren mysql-Images an, die über `registry.redhat.io` heruntergeladen werden können? (Wählen Sie zwei Antworten aus.)
 - a. podman search registry.redhat.io/mysql
 - b. podman images
 - c. podman pull mysql
 - d. podman search mysql

- ▶ 2. Welcher Befehl wird verwendet, um alle verfügbaren Image-Tags für das Container-Image httpd aufzuführen?
 - a. podman search httpd
 - b. podman images httpd
 - c. podman pull --all-tags=true httpd
 - d. Es steht kein Podman-Befehl zur Verfügung, um nach Tags zu suchen.

► **3. Mit welchen zwei Befehlen wird das httpd-Image mit dem Tag 2.4 abgerufen? (Wählen Sie zwei Antworten aus.)**

- 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. Welche Container-Images werden beim Ausführen der folgenden Befehle heruntergeladen?**

- 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 Für mysql wird kein Image heruntergeladen.
- d. quay.io/httpd:2.4 Für mysql wird kein Image heruntergeladen.

► Lösung

Verwenden von Registries

Wählen Sie anhand der folgenden Informationen die richtigen Antworten auf die folgenden Fragen aus:

Podman ist auf einem RHEL-Host mit dem folgenden Eintrag in der Datei /etc/containers/registries.conf verfügbar:

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

Die Hosts `registry.redhat.io` und `quay.io` verfügen über eine aktive Registry, weisen beide gültige Zertifikate auf und verwenden Version 1 der Registry. Die folgenden Images sind für jeden Host verfügbar:

Image-Namen/Tags pro Registry

Registry	Image
registry.redhat.io	rhel8/nginx/1.1.6
	rhel8/mysql-8.0
	rhel8/httpd-2.4
quay.io	mysql-5.7
	httpd-2.4

Es sind keine Images lokal verfügbar.

- 1. Welche zwei Befehle zeigen die verfügbaren mysql-Images an, die über `registry.redhat.io` heruntergeladen werden können? (Wählen Sie zwei Antworten aus.)
- podman search `registry.redhat.io/mysql`
 - podman images
 - podman pull mysql
 - podman search mysql
- 2. Welcher Befehl wird verwendet, um alle verfügbaren Image-Tags für das Container-Image `httpd` aufzuführen?
- podman search httpd
 - podman images httpd
 - podman pull --all-tags=true httpd
 - Es steht kein Podman-Befehl zur Verfügung, um nach Tags zu suchen.

► **3. Mit welchen zwei Befehlen wird das httpd-Image mit dem Tag 2.4 abgerufen? (Wählen Sie zwei Antworten aus.)**

- 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. Welche Container-Images werden beim Ausführen der folgenden Befehle heruntergeladen?**

- 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 Für mysql wird kein Image heruntergeladen.
- d. quay.io/httpd:2.4 Für mysql wird kein Image heruntergeladen.

Ändern von Container-Images

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Speichern und laden von Container-Images in lokale Dateien
- Löschen von Images aus dem lokalen Storage
- Erstellen neuer Container-Images anhand von Containern und Aktualisieren von Image-Metadaten
- Verwalten von Image-Tags für Bereitstellungszwecke

Einführung

Es gibt verschiedene Möglichkeiten, Image-Container unter Berücksichtigung von DevOps-Prinzipien zu verwalten. Beispiel: Ein Entwickler hat das Testen eines benutzerdefinierten Containers auf seinem Rechner fertiggestellt und muss dieses Container-Image für einen anderen Entwickler auf einen anderen Host oder einen Produktionsserver übertragen. Es stehen zwei Möglichkeiten zur Verfügung:

1. Speichern des Container-Images in einer .tar-Datei.
2. Veröffentlichen (Übertragen per *Push*) des Container-Images in einer Image-Registry.



Anmerkung

Eine der Möglichkeiten, auf die Entwickler diesen benutzerdefinierten Container erstellen können, wird weiter unten in diesem Kapitel (`podman commit`) erläutert. In den folgenden Kapiteln besprechen wir jedoch die empfohlene Vorgehensweise mit Containerfiles.

Speichern und Laden von Images

Bestehende Images aus dem lokalen Podman-Storage können mithilfe des Befehls `podman save` in einer .tar-Datei gespeichert werden. Die generierte Datei ist kein reguläres TAR-Archiv. Sie enthält Image-Metadaten und behält die ursprünglichen Image-Ebenen bei. Mit dieser Datei kann Podman das ursprüngliche Image wiederherstellen.

Die allgemeine Syntax des Unterbefehls `save` lautet wie folgt:

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

Podman sendet das generierte Image als Binärdaten an die Standardausgabe. Verwenden Sie die Option `-o`, um dies zu vermeiden.

Im folgenden Beispiel wird das zuvor aus dem Red Hat Container Catalog heruntergeladene MySQL-Container-Image in der Datei `mysql.tar` gespeichert:

Kapitel 4 | Verwalten von Container-Images

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

Beachten Sie die Verwendung der Option „-o“ in diesem Beispiel.

Verwenden Sie die durch den Unterbefehl `save` generierten `.tar`-Dateien zu Backup-Zwecken. Mit dem Befehl `podman load` kann das Container-Image wiederhergestellt werden. Die allgemeine Syntax des Befehls lautet wie folgt:

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

Der Befehl „`load`“ ist das Gegenteil des Befehls „`save`“.

Beispielsweise wird mit diesem Befehl ein Image geladen, das in einer Datei mit dem Namen `mysql.tar` gespeichert ist:

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

Wenn die als ein Argument angegebene `.tar`-Datei kein Container-Image mit Metadaten ist, schlägt der Befehl `podman load` fehl.



Anmerkung

Komprimieren Sie die vom Unterbefehl `save` generierte Datei mit Gzip mit dem Parameter `--compress`, um Speicherplatz zu sparen. Der Unterbefehl `load` verwendet den Befehl `gunzip`, bevor die Datei in den lokalen Storage importiert wird.

Löschen von Images

Podman speichert jedes heruntergeladene Image in seinem lokalen Storage, darunter auch Images, die derzeit durch keinen Container verwendet werden. Images können jedoch veraltet sein und sollten folglich ersetzt werden.



Anmerkung

Aktualisierungen an Images in einer Registry werden nicht automatisch aktualisiert. Das Image muss entfernt und dann erneut abgerufen werden, um zu gewährleisten, dass sich die aktuelle Version des Images im lokalen Storage befindet.

Führen Sie den Befehl `podman rmi` aus, um ein Image aus dem lokalen Storage zu löschen.

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

Auf ein Image kann für die Entfernung mithilfe seines Namens oder seiner ID verwiesen werden. Podman kann keine Images löschen, während Container dieses Image verwenden. Sie müssen alle Container, die dieses Image verwenden, stoppen und entfernen, bevor Sie sie löschen.

Um dies zu vermeiden, besitzt der Unterbefehl `rmi` die Option `--force`. Diese Option erzwingt das Entfernen eines Images, auch wenn das Image von mehreren Containern verwendet wird.

oder diese Container ausgeführt werden. Podman beendet und entfernt alle Container mit dem zwangsweise entfernten Image, bevor es entfernt wird.

Löschen sämtlicher Images

Verwenden Sie den folgenden Befehl, um sämtliche Images zu löschen, die nicht von Containern verwendet werden:

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

Mit diesem Befehl werden alle im lokalen Storage verfügbaren Image-IDs zurückgegeben und diese als Parameter an den Befehl `podman rmi` zur Entfernung weitergegeben. Images, die verwendet werden, werden nicht gelöscht. Dies schützt nicht verwendete Images jedoch nicht davor, dass sie entfernt werden.

Ändern von Images

Im Idealfall sollten alle Container-Images mit einem `Containerfile` erstellt werden, um einen übersichtlichen und schlanken Satz an Image-Ebenen ohne Protokolldateien, temporäre Dateien oder andere durch die Container-Anpassung erstellte Artefakte zu erstellen. Manche Benutzer stellen Container-Images jedoch möglicherweise im Ist-Zustand ohne ein `Containerfile` bereit. Ändern Sie als alternativen Ansatz zum Erstellen neuer Images einen laufenden Container vor Ort und speichern Sie dessen Ebenen, um ein neues Container-Image zu erstellen. Dieses Feature wird durch den Befehl `podman commit` bereitgestellt.



Warnung

Auch wenn der Befehl `podman commit` die einfachste Möglichkeit darstellt, neue Images zu erstellen, wird dieses Vorgehen aufgrund der Image-Größe (`commit` speichert die Protokolle und Prozess-ID-Dateien in den erfassten Ebenen) und der fehlenden Möglichkeit, Änderungen nachzuverfolgen, nicht empfohlen. Ein `Containerfile` ist ein robuster Mechanismus, um einen Container anhand eines visuell lesbaren Befehlssatzes anzupassen und zu ändern, ohne dass diese Dateien vom Betriebssystem generiert werden.

Der Befehl `podman commit` hat folgende Syntax:

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

Die folgende Tabelle enthält die wichtigsten Optionen, die für den Befehl `podman commit` zur Verfügung stehen:

Option	Beschreibung
<code>--author ""</code>	Identifiziert, wer das Container-Image erstellt hat.
<code>--message ""</code>	Enthält eine Commit-Meldung zur Registry.
<code>--format</code>	Wählt das Image-Format aus. Gültige Optionen sind <code>oci</code> und <code>docker</code> .



Anmerkung

Die Option `--message` ist im standardmäßigen OCI-Container-Format nicht verfügbar.

Führen Sie den Befehl `podman ps` aus, um nach der ID eines aktiven Containers in Podman zu suchen:

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

Letztendlich können Administratoren das Image anpassen und den Container auf den gewünschten Status festlegen. Um zu identifizieren, welche Dateien seit Container-Start geändert, erstellt oder gelöscht wurden, verwenden Sie den Unterbefehl `diff`. Für den Unterbefehl ist nur der Container-Name oder die Container-ID erforderlich:

```
[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
```

Der Unterbefehl `diff` kennzeichnet hinzugefügte Dateien mit einem A, geänderte Dateien mit einem C und gelöschte Dateien mit einem D.



Anmerkung

Der Befehl `diff` meldet dem Container-Dateisystem nur hinzugefügte, geänderte oder gelöschte Dateien. Dateien, die in einem ausgeführten Container gemountet werden, werden nicht als Teil des Container-Dateisystems betrachtet. Um die Liste der gemounteten Dateien und Verzeichnisse für einen ausgeführten Container abzurufen, verwenden Sie den Befehl `podman inspect`:

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

Jede Datei in dieser Liste oder Datei in einem Verzeichnis in dieser Liste wird nicht in der Ausgabe des Befehls `podman diff` angezeigt.

Führen Sie den folgenden Befehl aus, um die Änderungen für ein anderes Image durchzuführen:

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

Im vorherigen Beispiel wird ein neues Image mit dem Namen „mysql-custom“ erstellt.

Taggen von Images

Ein Projekt mit mehreren Images auf Grundlage derselben Software kann verteilt werden, wodurch einzelne Projekte für jedes Image erstellt werden. Für diesen Ansatz ist jedoch zusätzlicher Aufwand erforderlich, um die Images an den richtigen Standorten zu verwalten und bereitzustellen.

Container-Image-Registries unterstützen Tags, sodass sich mehrere Versionen desselben Projekts unterscheiden lassen. Beispielsweise verwendet ein Kunde möglicherweise ein mit einer MySQL- oder PostgreSQL-Datenbank ausgeführtes Container-Image mit einem Tag, mit dem unterschieden werden kann, welche Datenbank durch ein Container-Image verwendet wird.



Anmerkung

In der Regel werden die Tags von Container-Entwicklern verwendet, um zwischen mehreren Versionen derselben Software zu unterscheiden. Es werden mehrere Tags bereitgestellt, um eine Version einfach zu identifizieren. Auf der offiziellen MySQL-Container-Image-Website wird die Version als der Name des Tags (8.0) verwendet. Zudem kann dasselbe Image ein zweites Tag mit der Nebenversion aufweisen, um die Notwendigkeit zu minimieren, die neueste Version für eine spezifische Version abzurufen.

Mit dem Befehl `podman tag` können Sie ein Image mit einem Tag versehen:

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

Das Argument **IMAGE** ist der Image-Name mit einem optionalen Tag, der von Podman verwaltet wird. Das folgende Argument verweist auf den neuen alternativen Namen für das Image. Podman geht von der neuesten Version aus, die durch den Tag `latest` angegeben wird, wenn der Tag-Wert nicht angegeben ist. Zum Versehen eines Images mit einem Tag kann beispielsweise der folgende Befehl verwendet werden:

```
[user@host ~]$ podman tag mysql-custom devops/mysql
```

Die Option `mysql-custom` entspricht dem Image-Namen in der Container-Registry.

Wenn Sie stattdessen einen anderen Tagnamen verwenden möchten, sollten Sie den folgenden Befehl verwenden:

```
[user@host ~]$ podman tag mysql-custom devops/mysql:snapshot
```

Entfernen von Tags aus Images

Durch Ausführen des Befehls `podman rmi` können einem einzelnen Image mehrere Tags zugewiesen werden. Um sie zu entfernen, verwenden Sie wie zuvor erwähnt den Befehl `podman rmi`:

```
[user@host ~]$ podman rmi devops/mysql:snapshot
```



Anmerkung

Da mehrere Tags auf dasselbe Image verweisen können, sollte zum Entfernen eines Images, auf das mehrere Tags verweisen, zunächst jedes Tag einzeln entfernt werden.

Best Practices für das Taggen von Images

Das Tag `latest` wird automatisch von Podman hinzugefügt, wenn Sie kein Tag angeben, da in Podman angenommen wird, dass das Image der neueste Build ist. Dies ist jedoch in Abhängigkeit davon, wie Tags vom jeweiligen Projekt verwendet werden. Beispielsweise wird in den meisten Open-Source-Projekten `latest` als neueste Version erachtet und nicht der neueste Build.

Es werden jedoch mehrere Tags bereitgestellt, um die Notwendigkeit zu minimieren, die neueste Version einer bestimmten Version eines Projekts erneut aufzurufen. Wenn eine Projektversion, beispielsweise `2.1.10`, vorhanden ist, kann demnach ein anderes Tag mit der Bezeichnung `2.1` erstellt werden und über die Version `2.1.10` auf dasselbe Image verweisen. Das vereinfacht die Art, wie Images aus der Registry abgerufen werden.

Veröffentlichen von Images in einer Registry

Um ein Image in der Registry zu veröffentlichen, muss es sich im lokalen Storage von Podman befinden und zu Identifikationszwecken mit Tags versehen sein. Verwenden Sie zum Verschieben des Images in die Registry den Unterbefehl `push`:

```
[user@host ~]$ podman push [OPTIONS] IMAGE [DESTINATION]
```

Verwenden Sie beispielsweise den folgenden Befehl, um das Image `bitnami/nginx` an das zugehörige Repository zu übertragen:

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

Im vorherigen Beispiel wird das Image per Push an Quay.io übertragen.



Literaturhinweise

Podman-Site

<https://podman.io/>

► Angeleitete Übung

Erstellen eines benutzerdefinierten Apache-Container-Images

In dieser angeleiteten Übung erstellen Sie mit dem Befehl `podman commit` ein benutzerdefiniertes Apache-Container-Image.

Ergebnisse

Sie sollten in der Lage sein, ein benutzerdefiniertes Container-Image zu erstellen.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf dem Rechner `workstation` einen Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

- 1. Melden Sie sich bei Ihrem Quay.io-Benutzerkonto an, und starten Sie einen Container mit dem unter `quay.io/redhattraining/httpd-parent` verfügbaren Image. Mit der Option `-p` können Sie einen Port für die Weiterleitung angeben. In diesem Fall leitet Podman eingehende Anforderungen auf TCP-Port 8180 des Hosts an TCP-Port 80 des Containers weiter.

```
[student@workstation ~]$ podman login quay.io
Username: your_quay_username
Password: your_quay_password
Login Succeeded!
[student@workstation ~]$ podman run -d --name official-httpd \
> -p 8180:80 quay.io/redhattraining/httpd-parent
...output omitted...
Writing manifest to image destination
Storing signatures
`3a6baecaff2b` 4e8c53b026e04847dda5976b773ade1a3a712b1431d60ac5915d
```

Die letzte Zeile der Ausgabe unterscheidet sich von der oben gezeigten letzten Zeile. Notieren Sie sich die ersten zwölf Zeichen.

- 2. Erstellen Sie im Container `official-httpd` eine HTML-Seite.

- 2.1. Greifen Sie mit dem Befehl `podman exec` auf die Shell des Containers zu, und erstellen Sie eine HTML-Seite.

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

- 2.2. Beenden Sie den Container.

```
bash-4.4# exit
```

- 2.3. Überprüfen Sie mithilfe des Befehls `curl`, ob die HTML-Datei über den Rechner `workstation` erreichbar ist.

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

Die Ausgabe sollte wie folgt aussehen:

```
D0180 Page
```

- 3. Führen Sie den Befehl `podman diff` aus, um die Unterschiede im Container zwischen dem Image und der neuen Ebene zu untersuchen, die vom Container erstellt wurde.

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



Anmerkung

Oft wird in den Container-Images des Webservers das Verzeichnis `/var/www/html` als Volume bezeichnet. In diesen Fällen werden alle Dateien, die diesem Verzeichnis hinzugefügt werden, nicht als Teil des Container-Dateisystems betrachtet und nicht in der Ausgabe des Befehls `git diff` angezeigt. Das Container-Image `quay.io/redhattraining/httdp-parent` bezeichnet das Verzeichnis `/var/www/html` nicht als „Volume“. Daher wird der Wechsel zur Datei `/var/www/html/do180.html` als Wechsel zum zugrundeliegenden Container-Dateisystem betrachtet.

- 4. Erstellen Sie ein neues Image mit den Änderungen, die vom ausgeführten Container erstellt wurden.

- 4.1. Beenden Sie den Container `official-httdp`.

```
[student@workstation ~]$ podman stop official-httdp
`3a6baecaff2b`4e8c53b026e04847dda5976b773ade1a3a712b1431d60ac5915d
```

- 4.2. Führen Sie die Änderungen für ein neues Container-Image mit einem neuen Namen durch. Geben Sie für den Ersteller der Änderungen Ihren Namen an.

```
[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. Führen Sie die verfügbaren Container-Images auf:

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

Es wird in etwa die folgende Ausgabe erwartet:

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

Die Image-ID stimmt mit den ersten 12 Zeichen des Hashs überein. Die neuesten Images werden ganz oben aufgeführt.

► 5. Veröffentlichen Sie das gespeicherte Container-Image in der Container-Registry.

- 5.1. Laden Sie die Konfiguration Ihrer Kursumgebung.

Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

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

- 5.2. Führen Sie den folgenden Befehl aus, um das Image mit dem Registry-Hostnamen und -Tag zu kennzeichnen:

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

- 5.3. Führen Sie den Befehl `podman images` aus, um sicherzustellen, dass der neue Name dem Cache hinzugefügt wurde.

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

REPOSITORY	TAG	IMAGE ID	...
localhost/do180-custom-httdp	latest	31c3ac78e9d4	...
quay.io/\${RHT_OCP4_QUAY_USER}/do180-custom-httdp	v1.0	31c3ac78e9d4	...
quay.io/redhattraining/httpd-parent	latest	2cc07fbb5000	...

5.4. Veröffentlichen Sie das Image in ihrer Quay.io-Registry.

```
[student@workstation ~]$ podman push \
> quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
Getting image source signatures
Copying blob sha256:071d8bd765171080d01682844524be57ac9883e53079b6ac66707e19...
200.44 MB / 200.44 MB [=====] 1m38s
...output omitted...
Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb3...
2.99 KB / 2.99 KB [=====] 0s
Writing manifest to image destination
Copying config sha256:31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb3...
0 B / 2.99 KB [-----] 0s
Writing manifest to image destination
Storing signatures
```

**Anmerkung**

Wenn Sie das Image `do180-custom-httdp` übertragen, wird in Ihrem Quay.io-Konto ein privates Repository erstellt. Private Repositories werden derzeit von kostenlosen Quay.io-Tarifen nicht zugelassen. Sie können entweder das öffentliche Repository erstellen, bevor Sie das Image übertragen, oder das Repository anschließend als öffentlich festlegen.

5.5. Verifizieren Sie, dass das Image über Quay.io verfügbar ist. Damit der Befehl `podman search` ausgeführt werden kann, muss das Image durch Quay.io indiziert sein. Dies kann einige Stunden in Anspruch nehmen. Führen Sie daher den Befehl `podman pull` aus, um das Image abzurufen. Dies beweist, dass das Image verfügbar ist.

```
[student@workstation ~]$ podman pull \
> -q quay.io/${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
31c3ac78e9d4137c928da23762e7d32b00c428eb1036cab1caeeb3
```

▶ 6. Erstellen Sie anhand des neu veröffentlichten Images einen Container.

Verwenden Sie den Befehl `podman run`, um einen neuen Container zu starten. Verwenden Sie *Ihr_Quay_Benutzername/do180-custom-httdp:v1.0* als Basis-Image.

```
[student@workstation ~]$ podman run -d --name test-httdp -p 8280:80 \
> ${RHT_OCP4_QUAY_USER}/do180-custom-httdp:v1.0
c0f04e906bb12bd0e514cbd0e581d2746e04e44a468dfbc85bc29ffcc5acd16c
```

▶ 7. Greifen Sie mithilfe des Befehls `curl` auf die HTML-Seite zu. Achten Sie darauf, Port 8280 zu verwenden.

Die im vorherigen Schritt erstellte HTML-Seite sollte angezeigt werden.

```
[student@workstation ~]$ curl http://localhost:8280/do180.html
D0180 Page
```

Beenden

Führen Sie auf `workstation` das Skript `lab image-operations finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit ist die angeleitete Übung beendet.

► Praktische Übung

Verwalten von Images

Ergebnisse

Sie sollten in der Lage sein, ein benutzerdefiniertes Container-Image zu erstellen und Container-Images zu verwalten.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer student auf workstation ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Laden Sie mit dem Befehl `podman pull` das Container-Image `quay.io/redhattraining/nginx:1.17` herunter. Bei diesem Image handelt es sich um eine Kopie des unter `docker.io/library/nginx:1.17` verfügbaren Container-Images.
Stellen Sie sicher, dass Sie das Image erfolgreich heruntergeladen haben.
2. Starten Sie mithilfe des Nginx-Images einen neuen Container entsprechend den Spezifikationen in der folgenden Liste.
 - *Name*: `official-nginx`
 - *Als Daemon ausführen*: ja
 - *Container-Image*: `nginx`
 - *Portweiterleitung*: von Host-Port 8080 zu Container-Port 80
3. Melden Sie sich über den Befehl `podman exec` bei dem Container an. Ersetzen Sie die Inhalte der Datei `index.html` durch `D0180`. Das Webserver-Verzeichnis befindet sich unter `/usr/share/nginx/html`.
Verlassen Sie nach der Aktualisierung der Datei den Container, und führen Sie den Befehl `curl` aus, um auf die Webseite zuzugreifen.
4. Beenden Sie den ausgeführten Container, und bestätigen Sie die Änderungen, um ein neues Container-Image zu erstellen. Nennen Sie das neue Image `do180/mynginx` und fügen Sie als Tag `v1.0-SNAPSHOT` hinzu. Verwenden Sie die folgenden Spezifikationen:
 - *Image-Name*: `do180/mynginx`
 - *Image-Tag*: `v1.0-SNAPSHOT`
 - *Erstellername*: *Ihr Name*
5. Starten Sie mithilfe des aktualisierten Nginx-Images einen neuen Container entsprechend den Spezifikationen in der folgenden Liste.

- Name: official-nginx-dev
 - Als Daemon ausführen: ja
 - Container-Image: do180/mynginx:v1.0-SNAPSHOT
 - Portweiterleitung: von Host-Port 8080 zu Container-Port 80
6. Melden Sie sich mithilfe des Befehls `podman exec` beim Container an, um eine letzte Änderung vorzunehmen. Ersetzen Sie die Inhalte der Datei `/usr/share/nginx/html/index.html` durch D0180 Page.
Verlassen Sie nach der Aktualisierung der Datei den Container, und führen Sie den Befehl `curl` aus, um die Änderungen zu verifizieren.
7. Beenden Sie den ausgeführten Container, und bestätigen Sie die Änderungen, um das endgültige Container-Image zu erstellen. Nennen Sie das neue Image `do180/mynginx` und fügen Sie als Tag `v1.0` hinzu. Verwenden Sie die folgenden Spezifikationen:
- Image-Name: `do180/mynginx`
 - Image-Tag: `v1.0`
 - Erstellername: *Ihr Name*
8. Entfernen Sie das Entwicklungs-Image `do180/mynginx:v1.0-SNAPSHOT` aus dem lokalen Image-Storage.
9. Verwenden Sie das mit `do180/mynginx:v1.0` getaggte Image, um einen neuen Container mit folgenden Spezifikationen zu erstellen:
- Container-Name: `my-nginx`
 - Als Daemon ausführen: ja
 - Container-Image: `do180/mynginx:v1.0`
 - Portweiterleitung: ja, von Host-Port 8280 zu Container-Port 80

Verwenden Sie auf `workstation` den Befehl `curl`, um auf den Webserver zuzugreifen, der über Port 8280 zugänglich ist.

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem `workstation`-Rechner den Befehl `lab image-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf `workstation` den Befehl `lab image-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Verwalten von Images

Ergebnisse

Sie sollten in der Lage sein, ein benutzerdefiniertes Container-Image zu erstellen und Container-Images zu verwalten.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Laden Sie mit dem Befehl `podman pull` das Container-Image `quay.io/redhattraining/nginx:1.17` herunter. Bei diesem Image handelt es sich um eine Kopie des unter `docker.io/library/nginx:1.17` verfügbaren Container-Images.
Stellen Sie sicher, dass Sie das Image erfolgreich heruntergeladen haben.

- 1.1. Führen Sie den Befehl `podman pull` aus, um das Nginx-Container-Image abzurufen.

```
[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. Stellen Sie sicher, dass das Container-Image lokal verfügbar ist, indem Sie den Befehl `podman images` ausführen.

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

Mit diesem Befehl wird in etwa Folgendes ausgegeben:

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
quay.io/redhattraining/nginx	1.17	9beeba249f3e	6 months ago
			428MB

2. Starten Sie mithilfe des Nginx-Images einen neuen Container entsprechend den Spezifikationen in der folgenden Liste.

- *Name: official-nginx*
- *Als Daemon ausführen: ja*
- *Container-Image: nginx*

Kapitel 4 | Verwalten von Container-Images

- *Portweiterleitung:* von Host-Port 8080 zu Container-Port 80
- 2.1. Führen Sie auf workstation den Befehl `podman run` aus, um einen Container mit dem Namen `official-nginx` zu erstellen.

```
[student@workstation ~]$ podman run --name official-nginx \
> -d -p 8080:80 quay.io/redhattraining/nginx:1.17
b9d5739af239914b371025c38340352ac1657358561e7ebbd5472dfd5ff97788
```

3. Melden Sie sich über den Befehl `podman exec` bei dem Container an. Ersetzen Sie die Inhalte der Datei `index.html` durch `D0180`. Das Webserver-Verzeichnis befindet sich unter `/usr/share/nginx/html`.
- Verlassen Sie nach der Aktualisierung der Datei den Container, und führen Sie den Befehl `curl` aus, um auf die Webseite zuzugreifen.

- 3.1. Melden Sie sich über den Befehl `podman exec` bei dem Container an.

```
[student@workstation ~]$ podman exec -it official-nginx /bin/bash
root@b9d5739af239:/#
```

- 3.2. Aktualisieren Sie die Datei `index.html` unter `/usr/share/nginx/html`. Die Datei sollte `D0180` heißen.

```
root@b9d5739af239:/# echo 'D0180' > /usr/share/nginx/html/index.html
```

- 3.3. Beenden Sie den Container.

```
root@b9d5739af239:/# exit
```

- 3.4. Überprüfen Sie mithilfe des Befehls `curl`, ob die Datei `index.html` aktualisiert wurde.

```
[student@workstation ~]$ curl 127.0.0.1:8080
D0180
```

4. Beenden Sie den ausführten Container, und bestätigen Sie die Änderungen, um ein neues Container-Image zu erstellen. Nennen Sie das neue Image `do180/mynginx` und fügen Sie als Tag `v1.0-SNAPSHOT` hinzu. Verwenden Sie die folgenden Spezifikationen:

- Image-Name: `do180/mynginx`
- Image-Tag: `v1.0-SNAPSHOT`
- Erstellername: *Ihr Name*

- 4.1. Führen Sie den Befehl „`podman stop`“ aus, um den Container `official-nginx` zu beenden.

```
[student@workstation ~]$ podman stop official-nginx
b9d5739af239914b371025c38340352ac1657358561e7ebbd5472dfd5ff97788
```

- 4.2. Bestätigen Sie die Änderungen für ein neues Container-Image. Geben Sie für den Ersteller der Änderungen Ihren Namen an.

```
[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. Listen Sie die verfügbaren Container-Images auf, um das neu erstellte Image zu suchen.

```
[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. Starten Sie mithilfe des aktualisierten Nginx-Images einen neuen Container entsprechend den Spezifikationen in der folgenden Liste.

- *Name:* official-nginx-dev
- *Als Daemon ausführen:* ja
- *Container-Image:* do180/mynginx:v1.0-SNAPSHOT
- *Portweiterleitung:* von Host-Port 8080 zu Container-Port 80

- 5.1. Führen Sie auf workstation den Befehl `podman run` aus, um einen Container mit dem Namen `official-nginx-dev` zu erstellen.

```
[student@workstation ~]$ podman run --name official-nginx-dev \
> -d -p 8080:80 do180/mynginx:v1.0-SNAPSHOT
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a288661566671f0bbbb
```

6. Melden Sie sich mithilfe des Befehls `podman exec` beim Container an, um eine letzte Änderung vorzunehmen. Ersetzen Sie die Inhalte der Datei `/usr/share/nginx/html/index.html` durch `D0180 Page`.

Verlassen Sie nach der Aktualisierung der Datei den Container, und führen Sie den Befehl `curl` aus, um die Änderungen zu verifizieren.

- 6.1. Melden Sie sich über den Befehl `podman exec` bei dem Container an.

```
[student@workstation ~]$ podman exec -it official-nginx-dev /bin/bash
root@cfa21f02a77d:#
```

- 6.2. Aktualisieren Sie die Datei `index.html` unter `/usr/share/nginx/html`. Die Datei sollte `D0180 Page` lauten.

```
root@cfa21f02a77d:# echo 'D0180 Page' > /usr/share/nginx/html/index.html
```

- 6.3. Beenden Sie den Container.

Kapitel 4 | Verwalten von Container-Images

```
root@cfa21f02a77d:/# exit
```

- 6.4. Überprüfen Sie mithilfe des Befehls `curl`, ob die Datei `index.html` aktualisiert wurde.

```
[student@workstation ~]$ curl 127.0.0.1:8080
D0180 Page
```

7. Beenden Sie den ausführten Container, und bestätigen Sie die Änderungen, um das endgültige Container-Image zu erstellen. Nennen Sie das neue Image `do180/mynginx` und fügen Sie als Tag `v1.0` hinzu. Verwenden Sie die folgenden Spezifikationen:

- Image-Name: `do180/mynginx`
- Image-Tag: `v1.0`
- Erstellername: *Ihr Name*

- 7.1. Führen Sie den Befehl „`podman stop`“ aus, um den Container `official-nginx-dev` zu beenden.

```
[student@workstation ~]$ podman stop official-nginx-dev
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a28866156671f0bbbb
```

- 7.2. Bestätigen Sie die Änderungen für ein neues Container-Image. Geben Sie für den Ersteller der Änderungen Ihren Namen an.

```
[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. Listen Sie die verfügbaren Container-Images auf, um das neu erstellte Image zu suchen.

```
[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. Entfernen Sie das Entwicklungs-Image `do180/mynginx:v1.0-SNAPSHOT` aus dem lokalen Image-Storage.

- 8.1. Der gestoppte Container `official-nginx-dev` ist dennoch weiterhin vorhanden. Zeigen Sie den Container mit dem Befehl `podman ps` und dem Flag `-a` an.

```
[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. Entfernen Sie den Container mithilfe des Befehls `podman rm`.

```
[student@workstation ~]$ podman rm official-nginx-dev
cfa21f02a77d0e46e438c255f83dea2dfb89bb5aa72413a28866156671f0bbbb
```

8.3. Verifizieren Sie, dass der Container gelöscht wird, indem Sie denselben Befehl `podman ps` erneut senden.

```
[student@workstation ~]$ podman ps -a \
> --format="{{.ID}} {{.Names}} {{.Status}}"
b9d5739af239    official-nginx       Exited (0) 12 minutes ago
```

8.4. Führen Sie den Befehl „`podman rmi`“ aus, um das Image `do180/mynginx:v1.0-SNAPSHOT` zu entfernen.

```
[student@workstation ~]$ podman rmi do180/mynginx:v1.0-SNAPSHOT
Untagged: localhost/do180/mynginx:v1.0-SNAPSHOT
```

8.5. Verifizieren Sie, dass das Image nicht mehr vorhanden ist, indem Sie durch Ausführen des Befehls `podman images` alle Images auflisten.

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. Verwenden Sie das mit `do180/mynginx:v1.0` getaggte Image, um einen neuen Container mit folgenden Spezifikationen zu erstellen:

- Container-Name: `my-nginx`
- Als Daemon ausführen: ja
- Container-Image: `do180/mynginx:v1.0`
- Portweiterleitung: ja, von Host-Port 8280 zu Container-Port 80

Verwenden Sie auf `workstation` den Befehl `curl`, um auf den Webserver zuzugreifen, der über Port 8280 zugänglich ist.

9.1. Erstellen Sie mithilfe des Befehls „`podman run`“ den Container `my-nginx` gemäß den Spezifikationen.

```
[student@workstation ~]$ podman run -d --name my-nginx \
> -p 8280:80 do180/mynginx:v1.0
51958c8ec8d2613bd26f85194c66ca96c95d23b82c43b23b0f0fb9fded74da20
```

- 9.2. Überprüfen Sie mithilfe des Befehls `curl`, ob die Seite `index.html` verfügbar ist und den benutzerdefinierten Inhalt zurückgibt.

```
[student@workstation ~]$ curl 127.0.0.1:8280  
DO180 Page
```

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem `workstation`-Rechner den Befehl `lab image-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf `workstation` den Befehl `lab image-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Red Hat Container Catalog stellt unter `registry.redhat.io` getestete und zertifizierte Images bereit.
- Podman kann mit Remote-Container-Registries interagieren, um Container-Images zu suchen, abzurufen und zu verschieben.
- Image-Tags sind ein Mechanismus zum Unterstützen mehrerer Versionen eines Container-Images.
- Mit den von Podman bereitgestellten Befehlen können Container-Images im lokalen Storage und als komprimierte Dateien verwaltet werden.
- Verwenden Sie den Befehl `podman commit`, um ein Image aus einem Container zu erstellen.

Kapitel 5

Erstellen benutzerdefinierter Container-Images

Ziel

Entwerfen und Codieren eines Containerfiles zum Erstellen eines benutzerdefinierten Container-Images

Ziele

- Beschreiben der Ansätze zum Erstellen benutzerdefinierter Container-Images
- Erstellen eines Container-Images mit allgemeinen Containerfile-Befehlen

Abschnitte

- Konzipieren von benutzerdefinierten Container-Images (und Test)
- Erstellen benutzerdefinierter Container-Images mit Containerfiles (und angeleitete Übung)

Praktische Übung

- Erstellen benutzerdefinierter Container-Images

Konzipieren von benutzerdefinierten Container-Images

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Beschreiben der Ansätze zum Erstellen benutzerdefinierter Container-Images
- Suche nach vorhandenen Containerfiles, die als Startpunkt für die Erstellung benutzerdefinierter Container-Images verwendet werden sollen
- Definition der Bedeutung der Red Hat Software Collections Library (RHSCl) beim Entwerfen von Container-Images über die Red Hat-Registry
- Beschreibung der Source-to-Image (S2I)-Alternative zu Containerfiles

Erneute Verwendung vorhandener Containerfiles

Bei einer Methode zum Erstellen von Container-Images, die zuvor besprochen wurde, müssen Sie einen Container erstellen, ihn ändern, um die Anforderungen der Anwendung zu erfüllen, die darin ausgeführt werden soll, und dann die Änderungen in ein Image übernehmen. Diese Option ist zwar unkompliziert, jedoch nur für die Verwendung oder das Testen sehr spezifischer Änderungen geeignet. Es befolgt nicht die Best Practices für Software wie Wartbarkeit, Automatisierung der Entwicklung und Wiederholbarkeit.

Diese Einschränkungen werden durch Containerfiles behoben, eine weitere Option zum Erstellen von Container-Images. Containerfiles können einfach freigegeben, versionskontrolliert, wiederverwendet und erweitert werden.

Containerfiles erleichtern auch das Erweitern eines Images, dem sogenannten untergeordneten Image, aus einem anderen Image, dem sogenannten übergeordneten Image. Ein untergeordnetes Image enthält alles im übergeordneten Image sowie alle Änderungen und Ergänzungen, die zum Erstellen des Images vorgenommen wurden.



Anmerkung

Für den Rest dieses Kurses kann es sich bei einer Datei, die als Containerfile bezeichnet wird, um eine Datei mit dem Namen *Containerfile* oder *Dockerfile* handeln. Beide verwenden dieselbe Syntax. *Containerfile* ist der Standardname, der von OCI-konformen Tools verwendet wird.

Um Images gemeinsam zu nutzen und wiederzuverwenden, sind viele beliebte Anwendungen, Sprachen und Frameworks bereits in öffentlichen Image-Registries wie Quay.io [<https://quay.io>] verfügbar. Es ist durchaus hilfreich, Anwendungskonfigurationen den für Container empfohlenen bewährten Methoden entsprechend anzupassen. Wenn Sie ein bewährtes übergeordnetes Image als Grundlage verwenden, sparen Sie sich eine Menge Arbeit.

Die Verwendung eines hochwertigen übergeordneten Images erleichtert die Wartung, insbesondere dann, wenn das übergeordnete Image von dessen Ersteller aktualisiert wird, um Bugfixes und Sicherheitsprobleme zu berücksichtigen.

Kapitel 5 | Erstellen benutzerdefinierter Container-Images

Die Erstellung eines Containerfiles zur Erstellung eines untergeordneten Images aus einem bestehenden Container-Image wird häufig in den folgenden typischen Szenarien verwendet:

- Hinzufügen neuer Laufzeitbibliotheken wie Datenbank-Konnektoren
- Durchführen unternehmensweiter Anpassungen wie das Hinzufügen von SSL-Zertifikaten und Authentifizierungsanbietern
- Hinzufügen interner Librarys, die auf einer einzelnen Image-Ebene von mehreren Container-Images für verschiedene Anwendungen verwendet werden

Die Änderung eines bestehenden Containerfiles zur Erstellung eines neuen Images kann in anderen Szenarien ein geeigneter Ansatz sein. Beispiel:

- Abkürzen des Container-Image durch Entfernen nicht benötigter Materialien (zum Beispiel Manpages oder der Dokumentation unter /usr/share/doc).
- Sperren des übergeordneten Images bzw. einiger darin enthaltener Softwarepakete für eine bestimmte Version, um das Risiko bei künftigen Softwareaktualisierungen zu reduzieren

Docker Hub und die Red Hat Software Collections Library (RHSC) bieten Container-Images, die als übergeordnete Images oder zur Änderung der Containerfiles verwendet werden können.

Arbeiten mit der Red Hat Software Collections Library

Red Hat Software Collections Library (RHSC) – oder einfach nur Software Collections – ist die Lösung von Red Hat für Entwickler, welche die neuesten Entwicklungstools benötigen, was in der Regel mit dem Veröffentlichungszeitplan für RHEL nicht vereinbar ist.

Red Hat Enterprise Linux (RHEL) bietet eine stabile Umgebung für Unternehmensanwendungen. Dazu muss RHEL die Hauptversionen von Upstream-Paketen auf demselben Stand halten, um Formatänderungen bei der API und Konfigurationsdatei zu verhindern. Lösungen für Sicherheits- und Leistungsprobleme werden aus späteren Upstream-Versionen zurückportiert. Neue Features, welche die Rückwärtskompatibilität kompromittieren würden, werden jedoch nicht zurückportiert.

RHSC ermöglicht Softwareentwicklern die Verwendung der neuesten Version, ohne dass sich diese auf RHEL auswirkt. Dies liegt daran, dass die RHSC-Pakete die RHEL-Standardpakete nicht ersetzen oder mit ihnen konfigurieren. Die RHEL- und RHSC-Standardpakete werden parallel installiert.



Anmerkung

Alle RHEL-Abonnenten haben Zugriff auf die RHSC. Um eine bestimmte Software-Kollektion für einen bestimmten Benutzer bzw. eine bestimmte Anwendungsumgebung (z. B. MySQL 5.7, auch rh-mysql57 genannt) zu aktivieren, aktivieren Sie die Yum-Software-Repositorien der RHSC und führen Sie einige einfache Schritte aus.

Suche nach Containerfiles aus der Red Hat Software Collections Library

RHSC ist die umfassendste Quelle für Container-Images, die von der Red Hat-Image-Registry für Benutzer der RHEL Atomic Host und OpenShift Container Platform zur Nutzung zur Verfügung gestellt wird.

Red Hat stellt die RHSC-Containerfiles und die zugehörigen Quellen im `rhscl-dockerfiles`-Paket bereit, das über das RHSC-Repository zugänglich ist. Community-Benutzer können die

Containerfiles für die entsprechenden CentOS-basierten Container-Images aus dem GitHub-Repository unter <https://github.com/sclorg?q=-container> abrufen.



Anmerkung

Viele RHSCl-Container-Images bieten Unterstützung für Source-to-Image (S2I), was besser als OpenShift-Container-Plattform-Funktion bekannt ist. Supportanforderungen für S2I wirken sich nicht auf die Nutzung dieser Container-Images in Kombination mit Docker aus.

Container-Images in Red Hat Container Catalog (RHCC)

Für kritische Anwendungen sind vertrauenswürdige Container erforderlich. Red Hat Container Catalog ist ein Repository aus zuverlässigen, getesteten, zertifizierten und zu einer Kollektion zusammengestellten Container-Images, die auf Versionen von Red Hat Enterprise Linux (RHEL) und verbundenen Systemen aufgebaut sind. Über RHCC verfügbare Container-Images haben ein Qualitätssicherungsverfahren durchlaufen. Alle Komponenten wurden von Red Hat neu erstellt, um bekannte Sicherheitslücken zu vermeiden. Sie werden regelmäßig aktualisiert, sodass sie die erforderliche Version der Software auch dann bereits enthalten, wenn ein neues Images noch nicht verfügbar ist. Mit RHCC können Sie Images durchsuchen, bestimmte Images suchen und Informationen, wie z. B. Version, Inhalte und Verwendung, abrufen.

Suchen nach Images mit Quay.io

Quay.io ist ein erweitertes Container-Repository von CoreOS, das für die Teamzusammenarbeit optimiert wurde. Unter <https://quay.io/search> können Sie nach Container-Images suchen.

Durch Klicken auf den Namen eines Images erhalten Sie Zugriff auf die Image-Informationsseite, einschließlich des Zugriffs auf alle vorhandenen Tags für das Image und den Befehl zum Abrufen des Images.

Suche nach Containerfiles im Docker Hub

Achten Sie auf Images aus Docker Hub. Jeder kann ein Docker Hub-Konto erstellen und Container-Images darin veröffentlichen. Es gibt keine allgemeine Gewährleistung zur Qualität und Sicherheit; Images im Docker Hub können professionell entwickelt oder aber auch als einmaliges Experiment erstellt worden sein. Jedes Image muss individuell überprüft werden.

Nach einer Suchanfrage zu einem Image stellt die Dokumentationsseite unter Umständen einen Link zu dem zugehörigen Containerfile bereit. Beispiel: Bei der Suche nach mysql wird als erstes Ergebnis die Dokumentationsseite für das **offizielle MySQL-Image** angezeigt, das unter https://hub.docker.com/_/mysql verfügbar ist.

Auf dieser Seite verweist unter dem Abschnitt **Supported tags and respective Dockerfile links** jedes Tag auf das GitHub-Projekt **docker-library**, das Containerfiles für Images hostet, die mithilfe des automatischen Erstellvorgangs der Docker-Community erstellt wurden.

Die direkte URL für die MySQL 8.0-Containerfile-Baumstruktur von Docker Hub lautet <https://github.com/docker-library/mysql/blob/master/8.0>.

Beschreiben der Verwendung des OpenShift-Tools „Source-to-Image“

Source-to-Image (S2I) bietet eine alternative Methode zur Verwendung von Containerfiles für die Erstellung von Container-Images und kann entweder als Funktion über OpenShift oder als eigenständiges s2i-Programm verwendet werden. S2I ermöglicht es Entwicklern, bei ihrer Arbeit ihre gewohnten Tools zu benutzen. Sie sind nicht gezwungen, die Containerfile-Syntax zu erlernen oder Betriebssystembefehle wie yum zu verwenden. Das S2I-Dienstprogramm erstellt für gewöhnlich schlankere Images mit weniger Ebenen.

Bei Verwendung des S2I-Tools werden benutzerdefinierte Container-Images für Anwendungen anhand des folgenden Prozesses erstellt:

1. Starten Sie einen Container über ein Basis-Container-Image mit dem Namen „Builder-Image“. Dieses Image umfasst die Laufzeit für die Programmiersprache und grundlegende Entwicklungstools wie Compiler und Paketmanager.
2. Rufen Sie den Anwendungsquellcode ab (in der Regel von einem Git-Server) und senden Sie ihn an den Container.
3. Erstellen Sie die binären Anwendungsdateien im Container.
4. Speichern Sie nach der Bereinigung den Container als neues Container-Image, das die Laufzeit für die Programmiersprache und die binären Anwendungsdateien enthält.

Das Builder-Image ist ein herkömmliches Container-Image, das der Standard-Verzeichnisstruktur folgt und Skripts zur Verfügung stellt, die während des S2I-Prozesses aufgerufen werden. Die meisten dieser Builder-Images können auch außerhalb des S2I-Prozesses als Basis-Images für Containerfiles verwendet werden.

Der `s2i`-Befehl wird verwendet, um den S2I-Prozess außerhalb von OpenShift auszuführen (dies gilt nur für Docker-Umgebungen). Die Installation erfolgt auf RHEL-Systemen über das RPM-Paket `source-to-image` und auf anderen Plattformen – einschließlich Windows und MacOS – über die Installationsprogramme, die auf GitHub im S2I-Projekt verfügbar sind.



Literaturhinweise

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/>

RHSCl-Dockerfiles auf GitHub

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

Verwendung der Container-Images von Red Hat Software Collections

<https://access.redhat.com/articles/1752723>

Quay.io

<https://quay.io/search>

Docker Hub

<https://hub.docker.com/>

GitHub-Projekt "Docker Library"

<https://github.com/docker-library>

GitHub-Projekt "S2I"

<https://github.com/openshift/source-to-image>

► Quiz

Entwurfsansätze für Container-Images

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

► 1. Welche Methode zur Erstellung von Container-Images wird von der Container-Community empfohlen?

- a. Befehle in einem Basis-Betriebssystem-Container ausführen, den Container übermitteln und dann als neues Container-Image speichern oder exportieren
- b. Befehle über ein Containerfile ausführen und das erstellte Container-Image an die Image-Registry übermitteln
- c. Die Container-Image-Ebenen manuell über tar-Dateien erstellen
- d. Den Befehl `podman build` ausführen, um eine Container-Image-Beschreibung im YAML-Format zu verarbeiten

► 2. Welche beiden Vorteile bietet die Verwendung des eigenständigen S2I-Prozesses im Gegensatz zur Verwendung von Containerfiles? (Wählen Sie zwei Antworten aus.)

- a. Neben der Podman-Basiseinrichtung sind keine weiteren Tools erforderlich
- b. Es werden kleinere Container-Images mit weniger Ebenen erstellt
- c. Hochwertige Builder-Images werden wiederverwendet
- d. Das untergeordnete Image wird bei Änderungen des übergeordneten Image (zum Beispiel bei der Behebung von Sicherheitsproblemen) automatisch aktualisiert
- e. Es werden Images erstellt, die mit OpenShift kompatibel sind (im Gegensatz zu Container-Images, die über Docker-Tools erstellt werden).

► 3. Was sind zwei typische Szenarien für die Erstellung eines Containerfiles zum Erstellen eines untergeordneten Images von einem vorhandenen Image? (Wählen Sie zwei Antworten aus.)

- a. Hinzufügen einer neuen Laufzeitbibliothek
- b. Festlegen von Beschränkungen für den Zugriff eines Containers auf die CPU des Host-Rechners
- c. Hinzufügen interner Librarys, die auf einer einzelnen Image-Ebene von mehreren Container-Images für verschiedene Anwendungen verwendet werden

► Lösung

Entwurfsansätze für Container-Images

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

► 1. Welche Methode zur Erstellung von Container-Images wird von der Container-Community empfohlen?

- a. Befehle in einem Basis-Betriebssystem-Container ausführen, den Container übermitteln und dann als neues Container-Image speichern oder exportieren
- b. Befehle über ein Containerfile ausführen und das erstellte Container-Image an die Image-Registry übermitteln
- c. Die Container-Image-Ebenen manuell über tar-Dateien erstellen
- d. Den Befehl podman build ausführen, um eine Container-Image-Beschreibung im YAML-Format zu verarbeiten

► 2. Welche beiden Vorteile bietet die Verwendung des eigenständigen S2I-Prozesses im Gegensatz zur Verwendung von Containerfiles? (Wählen Sie zwei Antworten aus.)

- a. Neben der Podman-Basiseinrichtung sind keine weiteren Tools erforderlich
- b. Es werden kleinere Container-Images mit weniger Ebenen erstellt
- c. Hochwertige Builder-Images werden wiederverwendet
- d. Das untergeordnete Image wird bei Änderungen des übergeordneten Image (zum Beispiel bei der Behebung von Sicherheitsproblemen) automatisch aktualisiert
- e. Es werden Images erstellt, die mit OpenShift kompatibel sind (im Gegensatz zu Container-Images, die über Docker-Tools erstellt werden).

► 3. Was sind zwei typische Szenarien für die Erstellung eines Containerfiles zum Erstellen eines untergeordneten Images von einem vorhandenen Image? (Wählen Sie zwei Antworten aus.)

- a. Hinzufügen einer neuen Laufzeitbibliothek
- b. Festlegen von Beschränkungen für den Zugriff eines Containers auf die CPU des Host-Rechners
- c. Hinzufügen interner Librarys, die auf einer einzelnen Image-Ebene von mehreren Container-Images für verschiedene Anwendungen verwendet werden

Erstellen benutzerdefinierter Container-Images mit Containerfiles

Ziele

Nach Abschluss dieses Abschnitts sollten Kursteilnehmer mit benutzerdefinierten Containerfile-Befehlen Container-Images erstellen können.

Erstellen von Basis-Containern

Ein Containerfile ist ein Mechanismus zum Automatisierung der Erstellung von Container-Images. Der Erstellungsprozess für Images aus einem Containerfile besteht aus den folgenden drei Schritten:

1. Erstellen eines Arbeitsverzeichnisses
2. Schreiben des Containerfiles
3. Erstellen des Images mit Podman

Erstellen eines Arbeitsverzeichnisses

Das Arbeitsverzeichnis ist das Verzeichnis, das alle Dateien enthält, die zum Erstellen des Images erforderlich sind. Um zu verhindern, dass unnötige Dateien in das Image eingeschlossen werden, empfiehlt es sich, ein leeres Arbeitsverzeichnis zu erstellen. Aus Sicherheitsgründen sollte das Root-Verzeichnis / niemals als Arbeitsverzeichnis für Image-Builds verwendet werden.

Schreiben der Containerfile-Spezifikation

Ein Containerfile ist eine Textdatei mit dem Namen *Containerfile* oder *Dockerfile*, die im Arbeitsverzeichnis vorhanden sein muss. Diese Datei enthält die Anweisungen zum Erstellen des Images. Die grundlegende Syntax eines Containerfiles folgt:

```
# Comment  
INSTRUCTION arguments
```

Zeilen, die mit einem Nummernzeichen (#) beginnen, sind Kommentare. *INSTRUCTION* steht für ein beliebiges Containerfile-Anweisungs-Keyword. Bei Anweisungen wird die Groß-/ Kleinschreibung nicht beachtet. Üblicherweise werden Anweisungen jedoch großgeschrieben, um die Sichtbarkeit zu verbessern.

Die erste Anweisung ohne Kommentare muss eine FROM-Anweisung sein, um das Basis-Image zu bestimmen. Containerfile-Anweisungen werden mit diesem Image in einen neuen Container ausgeführt und dann in ein neues Image übernommen. Die nächste (ggf. vorhandene) Anweisung wird in das neue Image eingefügt. Die Ausführungsreihenfolge der Anweisungen entspricht der Reihenfolge ihres Auftretens in dem Containerfile.



Anmerkung

Die ARG-Anweisung kann vor der FROM-Anweisung erscheinen. ARG-Anweisungen werden jedoch in diesem Abschnitt nicht beschrieben.

Kapitel 5 | Erstellen benutzerdefinierter Container-Images

Jede Containerfile-Anweisung wird in einem unabhängigen Container ausgeführt, wobei ein aus jedem vorherigen Befehl erstelltes Zwischen-Image verwendet wird. Dies bedeutet, dass jede Anweisung unabhängig von anderen Anweisungen in dem Containerfile ist. Im Folgenden sehen Sie ein Beispiel-Containerfile zur Erstellung eines einfachen Apache-Webserver-Containers:

```
# This is a comment line ①
FROM ubi8/ubi:8.3 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 Zeilen, die mit einem Nummernzeichen (#) beginnen, sind Kommentare.
- ② Die FROM-Anweisung deklariert, dass das neue Container-Image das ubi8/ubi:8.3-Container-Basis-Image erweitert. Containerfiles können jedes andere Container-Image als Basis-Image verwenden, nicht nur Images aus Betriebssystem-Distributionen. Red Hat bietet eine Reihe an zertifizierten und getesteten Container-Images und empfiehlt dringend, diese Container-Images als Grundlage zu verwenden.
- ③ Mit LABEL werden generische Metadaten zu einem Image hinzugefügt. Ein LABEL ist ein einfaches Schlüssel-Wert-Paar.
- ④ MAINTAINER gibt das Feld Author der Metadaten des generierten Container-Images an. Sie können mithilfe des Befehls podman inspect Image-Metadaten anzeigen.
- ⑤ RUN führt Befehle auf einer neuen, dem aktuellen Image übergeordnete Ebene aus. Die zum Ausführen von Befehlen verwendete Shell ist /bin/sh.
- ⑥ EXPOSE zeigt an, dass der Container zur Laufzeit den angegebenen Netzwerkport überwacht. Die Anweisung EXPOSE definiert nur Metadaten; sie ermöglicht nicht den Zugriff auf Ports über den Host. Die Option -p im Befehl podman run macht Container-Ports vom Host verfügbar.
- ⑦ Mit ENV werden Umgebungsvariablen definiert, die für den Container verfügbar sind. Sie können in dem Containerfile mehrere ENV-Anweisungen deklarieren. Sie können mithilfe des Befehls env im Container alle Umgebungsvariablen anzeigen.
- ⑧ Die Anweisung ADD kopiert Dateien oder Ordner aus einer lokalen oder Remote-Quelle und fügt sie zum Container-Dateisystem hinzu. Wenn Sie lokale Dateien kopieren, müssen sich diese im Arbeitsverzeichnis befinden. Die ADD-Anweisung entpackt lokale .tar-Dateien in das Image-Zielverzeichnis.
- ⑨ COPY kopiert Dateien aus dem Arbeitsverzeichnis und fügt sie dem Dateisystem des Containers hinzu. Mit dieser Containerfile-Anweisung ist es nicht möglich, eine remote Datei mithilfe der URL zu kopieren.
- ⑩ USER gibt den Benutzernamen oder die UID an, die verwendet werden soll, wenn das Container-Image für die Anweisungen RUN, CMD und ENTRYPOINT ausgeführt werden soll.

Es empfiehlt sich aus Sicherheitsgründen, einen anderen Benutzer als den `root`-Benutzer zu definieren.

- ⑪ `ENTRYPOINT` gibt den Standardbefehl an, der ausgeführt werden soll, wenn das Image in einem Container ausgeführt wird. Wenn nichts angegeben ist, entspricht `/bin/sh -c` dem standardmäßigen `ENTRYPOINT`.
- ⑫ `CMD` stellt die Standardargumente für die `ENTRYPOINT`-Anweisung bereit. Wenn der standardmäßige `ENTRYPOINT` (`/bin/sh -c`) anwendet, bildet `CMD` einen ausführbaren Befehl und Parameter, die beim Start des Containers ausgeführt werden.

CMD und ENTRYPOINT

`ENTRYPOINT`- und `CMD`-Anweisungen kommen in zwei Formaten vor:

- Exec-Format (mit einem JSON-Array):

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

- Shell-Format:

```
ENTRYPOINT command param1 param2
CMD param1 param2
```

Exec ist das bevorzugte Format. Das Shell-Format umschließt die Befehle in einer `/bin/sh -c`-Shell, wodurch ein manchmal unnötiger Shell-Prozess entsteht. Einige Kombinationen sind ebenfalls nicht zulässig oder funktionieren möglicherweise nicht wie erwartet. Wenn beispielsweise `ENTRYPOINT` dem `["ping"]` (exec-Format) und `CMD` dem `localhost` (Shell-Format) entspricht, lautet der erwartete ausgeführte Befehl `ping localhost`, der Container versucht jedoch, den ungültigen Befehl `ping /bin/sh -c localhost` auszuführen.

Das Containerfile sollte maximal eine `ENTRYPOINT`- und eine `CMD`-Anweisung enthalten. Wenn mehr als eine Anweisung vorhanden ist, wird nur die letzte wirksam. `CMD` kann vorhanden sein, ohne dass ein `ENTRYPOINT` angegeben wird. In diesem Fall wird der `ENTRYPOINT` des Basis-Images oder der standardmäßige `ENTRYPOINT` angewendet, wenn nichts angegeben ist.

Podman kann die `CMD`-Anweisung beim Starten eines Containers überschreiben. Falls vorhanden, bilden alle Parameter für den `podman run`-Befehl nach dem Image-Namen die `CMD`-Anweisung. Beispielsweise führt die folgende Anweisung dazu, dass der aktive Container die aktuelle Zeit anzeigt.

```
ENTRYPOINT ["/bin/date", "+%H:%M"]
```

Der `ENTRYPOINT` definiert den auszuführenden Befehl und die Parameter. Daher kann die `CMD`-Anweisung nicht verwendet werden. Im folgenden Beispiel wird dieselbe Funktion wie die vorherige dargestellt – mit dem zusätzlichen Vorteil, dass die `CMD`-Anweisung beim Starten eines Containers überschrieben werden kann:

```
ENTRYPOINT ["/bin/date"]
CMD ["+%H:%M"]
```

Wenn ein Container ohne Angabe eines Parameters gestartet wird, wird in beiden Fällen die aktuelle Zeit angezeigt:

```
[student@workstation ~]$ sudo podman run -it do180/rhel  
11:41
```

Wenn ein Parameter im zweiten Fall nach dem Image-Namen im Befehl `podman run` angegeben wird, wird die CMD-Anweisung überschrieben. Mit dem folgenden Befehl wird der aktuelle Wochentag anstatt der Zeit angezeigt:

```
[student@workstation demo-basic]$ sudo podman run -it do180/rhel +%A  
Tuesday
```

Ein anderer Ansatz besteht darin, den standardmäßigen ENTRYPPOINT und die CMD-Anweisung zu verwenden, um den ursprünglichen Befehl zu definieren. Die folgende Anweisung zeigt die aktuelle Zeit an, allerdings mit dem Vorteil, dass sie während der Laufzeit überschrieben werden kann.

```
CMD ["date", "+%H:%M"]
```

ADD und COPY

ADD- und COPY-Anweisungen kommen in zwei Formaten vor:

- Das Shell-Format:

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

- Das Exec-Formular:

```
ADD ["<_source_ >", ... "<_destination_ >"]  
COPY ["<_source_ >", ... "<_destination_>"]
```

Wenn die Quelle ein Dateisystempfad ist, muss sie sich im Arbeitsverzeichnis befinden.

Mit der ADD-Anweisung können Sie ebenfalls eine Ressource mithilfe einer URL angeben.

```
ADD http://someserver.com/filename.pdf /var/www/html
```

Wenn die Quelle eine komprimierte Datei ist, dekomprimiert die ADD-Anweisung die Datei im Ziel-Ordner. Die COPY-Anweisung bietet diese Funktion nicht.



Warnung

Bei Verwendung der ADD- und COPY-Anweisung werden die Dateien kopiert. Dabei werden die Berechtigungen und der root-Benutzer als Eigentümer beibehalten, auch wenn die USER-Anweisung angegeben ist. Red Hat empfiehlt, nach dem Kopieren eine RUN-Anweisung zu verwenden, um den Eigentümer zu ändern und Fehler infolge einer abgelehnten Berechtigung zu vermeiden.

Überlagern von Images

Jede Anweisung in einem Containerfile erstellt eine neue Image-Ebene. Zu viele Anweisungen in dem Containerfile bedingen ein Übermaß an Ebenen, wodurch die Images sehr umfangreich werden. Betrachten Sie beispielsweise die folgende RUN-Anweisung in einem Containerfile:

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

Dieses Beispiel zeigt die Erstellung eines Container-Images mit drei Ebenen (eine für jede RUN-Anweisung). Red Hat empfiehlt, die Anzahl der Ebenen zu minimieren. Sie können dasselbe Ziel erreichen, indem Sie eine einzelne Ebene mit der &&-Konjunktion in der RUN-Anweisung erstellen.

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

Das Problem bei diesem Ansatz ist, dass die Lesbarkeit des Containerfiles abnimmt. Verwenden Sie den \-Escape-Code, um Zeilenumbrüche einzufügen und die Lesbarkeit zu verbessern. Sie können auch Zeilen einrücken, um die Befehle auszurichten:

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

In diesem Beispiel wird nur eine Ebene erstellt, und die Lesbarkeit wird verbessert. RUN-, COPY- und ADD-Anweisungen erstellen neue Image-Ebenen, RUN kann jedoch auf diese Weise verbessert werden.

Red Hat empfiehlt, ähnliche Formatierungsregeln auf andere Anweisungen anzuwenden, die mehrere Parameter akzeptieren, beispielsweise LABEL und 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"
```

Erstellen von Images mit Podman

Der podman build-Befehl verarbeitet das Containerfile und erstellt ein neues Image auf Grundlage der darin enthaltenen Anweisungen. Der Befehl hat folgende Syntax:

```
$ podman build -t NAME:TAG DIR
```

DIR ist der Pfad zum Arbeitsverzeichnis, das das Containerfile enthalten muss. Dies kann das aktuelle Arbeitsverzeichnis sein, das durch einen Punkt (.) angegeben wird, wenn das Arbeitsverzeichnis das aktuelle Arbeitsverzeichnis ist. *NAME:TAG* ist ein Name mit einem Tag, der dem neuen Image zugewiesen wird. Wenn das *TAG* nicht angegeben wird, dann wird das Image automatisch als *latest* markiert.



Literaturhinweise

Dockerfile-Referenzhandbuch

<https://docs.docker.com/engine/reference/builder/>

Erstellen von Basis-Images

<https://docs.docker.com/engine/userguide/eng-image/baseimages/>

► Angeleitete Übung

Erstellen eines Apache-Basis-Container-Images

In dieser Übung erstellen Sie ein Apache-Basis-Container-Image.

Ergebnisse

Sie sollten in der Lage sein, ein benutzerdefiniertes Apache-Container-Image zu erstellen, das auf einem Red Hat Universal Base Image 8-Image basiert.

Bevor Sie Beginnen

Führen Sie den folgenden Befehl aus, um die relevanten Lab-Dateien herunterzuladen und um sicherzustellen, dass Docker ausgeführt wird:

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

Anweisungen

► 1. Erstellen Sie das Apache-Containerfile.

- 1.1. Öffnen Sie ein Terminal auf `workstation`. Erstellen Sie mit Ihrem bevorzugten Editor ein neues Containerfile.

```
[student@workstation  
~]$ vim /home/student/D0180/labs/dockerfile-create/containerfile
```

- 1.2. Verwenden Sie UBI 8.3 als Basis-Image, indem Sie dem neuen Containerfile oben die folgende `FROM`-Anweisung hinzufügen:

```
FROM ubi8/ubi:8.3
```

- 1.3. Fügen Sie unterhalb der `FROM`-Anweisung die `MAINTAINER`-Anweisung hinzu, um das `Author`-Feld im neuen Image festzulegen. Ersetzen Sie die Werte durch Ihren Namen und Ihre E-Mail-Adresse.

```
MAINTAINER Your Name <_youremail_>
```

- 1.4. Fügen Sie unterhalb der `MAINTAINER`-Anweisung die folgende `LABEL`-Anweisung hinzu, um dem neuen Image Beschreibungsmetadaten hinzuzufügen:

```
LABEL description="A custom Apache container based on UBI 8"
```

- 1.5. Fügen Sie mithilfe des `yum install`-Befehls eine `RUN`-Anweisung hinzu, um Apache auf dem neuen Container zu installieren.

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

- 1.6. Fügen Sie eine RUN-Anweisung hinzu, um die Inhalte der standardmäßigen HTTPD-Startseite zu ersetzen.

```
RUN echo "Hello from Containerfile" > /var/www/html/index.html
```

- 1.7. Verwenden Sie die EXPOSE-Anweisung unterhalb der RUN-Anweisung, um den Port zu dokumentieren, den der Container während der Laufzeit überwacht. Setzen Sie in dieser Instanz den Port auf 80, da dies die Standardeinstellung für Apache-Server ist.

```
EXPOSE 80
```



Anmerkung

Die EXPOSE-Anweisung macht den angegebenen Port jedoch nicht für den Host verfügbar. Stattdessen fungiert die Anweisung als Metadaten dahingehend, welche Ports der Container überwacht.

- 1.8. Verwenden Sie am Dateiende die folgende CMD-Anweisung, um httpd als standardmäßigen Entry Point festzulegen:

```
CMD ["httpd", "-D", "FOREGROUND"]
```

- 1.9. Vergewissern Sie sich, dass Ihr Containerfile dem folgenden entspricht, bevor Sie es speichern und mit den nächsten Schritten fortfahren:

```
FROM ubi8/ubi:8.3

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. Erstellen und überprüfen Sie das Apache-Container-Image.

- 2.1. Verwenden Sie die folgenden Befehle, um ein Basis-Container-Image von Apache anhand der neu erstellten Containerfile-Datei zu erstellen:

```
[student@workstation ~]$ cd /home/student/D0180/labs/dockerfile-create
[student@workstation dockerfile-create]$ podman build --layers=false \
> -t do180/apache .
STEP 1: FROM ubi8/ubi:8.3
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 Das in der FROM-Anweisung aufgelistete Container-Image wird nur heruntergeladen, wenn es nicht bereits im lokalen Storage vorhanden ist.



Anmerkung

Podman erstellt während des Erstellungsprozesses viele anonyme Zwischen-Images. Sie werden nur aufgeführt, wenn -a verwendet wird. Verwenden Sie die Option --layers=false des Unterbefehls build, um Podman anzuweisen, Zwischen-Images zu löschen.

- 2.2. Führen Sie nach der Erstellung den Befehl `podman images` aus, um das neue Image im Image-Repository anzuzeigen.

REPOSITORY	TAG	IMAGE ID	CREATED
SIZE			
localhost/do180/apache	latest	8ebfe343e08c	15 seconds ago
MB			234
registry.access.redhat.com/ubi8/ubi	8.3	4199acc83c6a	6 weeks ago
MB			213

- 3. Führen Sie den Apache-Container aus.

- 3.1. Verwenden Sie den folgenden Befehl, um unter Verwendung des Apache-Images einen Container auszuführen:

Kapitel 5 | Erstellen benutzerdefinierter Container-Images

```
[student@workstation dockerfile-create]$ podman run --name lab-apache \
> -d -p 10080:80 do180/apache
fa1d1c450e8892ae085dd8bbf763edac92c41e6ffaa7ad6ec6388466809bb391
```

3.2. Führen Sie den Befehl `podman ps` aus, um den ausgeführten Container anzuzeigen.

```
[student@workstation dockerfile-create]$ podman ps
CONTAINER ID IMAGE COMMAND ...output omitted...
fa1d1c450e88 localhost/do180/apache:latest httpd -D FOREGROU...output omitted...
```

3.3. Prüfen Sie anhand des Befehls `curl`, ob der Server ausgeführt wird.

```
[student@workstation dockerfile-create]$ curl 127.0.0.1:10080
Hello from Containerfile
```

Beenden

Führen Sie auf `workstation` das Skript `lab dockerfile-create finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit ist die angeleitete Übung beendet.

► Praktische Übung

Erstellen benutzerdefinierter Container-Images

In dieser praktischen Übung erstellen Sie ein Containerfile, um ein benutzerdefiniertes Apache Webserver-Container-Image zu erstellen. Das benutzerdefinierte Image basiert auf einem RHEL 8.3-UBI-Image und stellt eine benutzerdefinierte `index.html`-Seite bereit.

Ergebnisse

Sie sollten in der Lage sein, einen benutzerdefinierten Apache Webserver-Container zu erstellen, der statische HTML-Dateien hostet.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Überprüfen Sie den bereitgestellten Containerfile-Stub im Ordner `/home/student/DO180/labs/dockerfile-review/`. Bearbeiten Sie das `Containerfile`, und stellen Sie sicher, dass es die folgenden Spezifikationen erfüllt:
 - Das Basis-Image ist `ubi8/ubi:8.3`
 - Legt den gewünschten Erstellernamen und die E-Mail-ID mit der Anweisung `MAINTAINER` fest
 - Legt die Umgebungsvariable `PORT` auf 8080 fest
 - Installieren Sie Apache (`httpd`-Paket).
 - Ändern Sie die Apache-Konfigurationsdatei `/etc/httpd/conf/httpd.conf` so, dass anstelle des Standardports 80 der Port 8080 überwacht wird.
 - Ändern Sie den Eigentümer der Ordner `/etc/httpd/logs` und `/run/httpd` in den Benutzer bzw. die Gruppe `apache` (der UID- und GID-Wert ist 48).
 - Stellen Sie den in der Umgebungsvariablen `PORT` festgelegten Wert bereit, um Container-Benutzern zu zeigen, wie sie auf den Apache Webserver zugreifen können.
 - Kopieren Sie die Inhalte des Ordners `src/` im Lab-Verzeichnis in die Apache DocumentRoot-Datei (`/var/www/html/`) im Container.

Der Ordner `src` enthält eine einzelne `index.html`-Datei, mittels derer die Nachricht `Hello World!` ausgegeben wird.

- Starten Sie den Apache-Daemon `httpd` im Vordergrund mithilfe des folgenden Befehls:

```
httpd -D FOREGROUND
```

2. Erstellen Sie das benutzerdefinierte Apache-Image mit dem Namen do180/custom-apache.
3. Erstellen Sie einen neuen Container im getrennten Modus mit den folgenden Merkmalen:
 - Name: containerfile
 - Container-Image: do180/custom-apache
 - Portweiterleitung: von Host-Port 20080 zu Container-Port 8080
 - Als Daemon ausführen: jaÜberprüfen Sie, ob der Container bereit ist und ausgeführt wird.
4. Verifizieren Sie, dass der Server die HTML-Datei bereitstellt.

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem workstation-Rechner den Befehl `lab dockerfile-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf workstation den Befehl `lab dockerfile-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Erstellen benutzerdefinierter Container-Images

In dieser praktischen Übung erstellen Sie ein Containerfile, um ein benutzerdefiniertes Apache Webserver-Container-Image zu erstellen. Das benutzerdefinierte Image basiert auf einem RHEL 8.3-UBI-Image und stellt eine benutzerdefinierte `index.html`-Seite bereit.

Ergebnisse

Sie sollten in der Lage sein, einen benutzerdefinierten Apache Webserver-Container zu erstellen, der statische HTML-Dateien hostet.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Überprüfen Sie den bereitgestellten Containerfile-Stub im Ordner `/home/student/DO180/labs/dockerfile-review/`. Bearbeiten Sie das `Containerfile`, und stellen Sie sicher, dass es die folgenden Spezifikationen erfüllt:
 - Das Basis-Image ist `ubi8/ubi:8.3`
 - Legt den gewünschten Erstellernamen und die E-Mail-ID mit der Anweisung `MAINTAINER` fest
 - Legt die Umgebungsvariable `PORT` auf 8080 fest
 - Installieren Sie Apache (`httpd`-Paket).
 - Ändern Sie die Apache-Konfigurationsdatei `/etc/httpd/conf/httpd.conf` so, dass anstelle des Standardports 80 der Port 8080 überwacht wird.
 - Ändern Sie den Eigentümer der Ordner `/etc/httpd/logs` und `/run/httpd` in den Benutzer bzw. die Gruppe `apache` (der UID- und GID-Wert ist 48).
 - Stellen Sie den in der Umgebungsvariablen `PORT` festgelegten Wert bereit, um Container-Benutzern zu zeigen, wie sie auf den Apache Webserver zugreifen können.
 - Kopieren Sie die Inhalte des Ordners `src/` im Lab-Verzeichnis in die Apache DocumentRoot-Datei (`/var/www/html/`) im Container.

Der Ordner `src` enthält eine einzelne `index.html`-Datei, mittels derer die Nachricht `Hello World!` ausgegeben wird.

- Starten Sie den Apache-Daemon `httpd` im Vordergrund mithilfe des folgenden Befehls:

Kapitel 5 | Erstellen benutzerdefinierter Container-Images

```
httpd -D FOREGROUND
```

- 1.1. Verwenden Sie Ihren bevorzugten Editor, um das Containerfile zu ändern, das sich im Ordner /home/student/D0180/labs/dockerfile-review/ befindet.

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

- 1.2. Legen Sie das Basis-Image für das Containerfile auf ubi8/ubi:8.3 fest.

```
FROM ubi8/ubi:8.3
```

- 1.3. Legen Sie Ihren Namen und Ihre E-Mail-Adresse mit einer MAINTAINER-Anweisung fest.

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

- 1.4. Erstellen Sie eine Umgebungsvariable namens PORT, und setzen Sie sie auf 8080.

```
ENV PORT 8080
```

- 1.5. Installieren Sie den Apache-Server.

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

- 1.6. Ändern Sie die Konfigurationsdatei für den Apache HTTP-Server so, dass der Port 8080 überwacht wird, und ändern Sie den Eigentümer der Server-Arbeitsordner mit einer einzelnen RUN-Anweisung.

```
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. Führen Sie den Container unter Verwendung des Benutzernamens apache und der USER-Anweisung aus. Verwenden Sie die EXPOSE-Anweisung, um den Port zu dokumentieren, den der Container während der Laufzeit abhört. Setzen Sie in diesem Fall den Port auf die PORT-Umgebungsvariable. Dies ist die Standardeinstellung für Apache-Server.

```
USER apache  
  
# Expose the custom port that you provided in the ENV var  
EXPOSE ${PORT}
```

- 1.8. Kopieren Sie alle Dateien aus dem Ordner src in den Apache DocumentRoot-Pfad unter /var/www/html.

```
# Copy all files under src/ folder to Apache DocumentRoot (/var/www/html)
COPY ./src/ /var/www/html/
```

- 1.9. Fügen Sie schließlich eine CMD-Anweisung hinzu, um httpd im Vordergrund auszuführen, und speichern Sie dann das Containerfile.

```
# Start Apache in the foreground
CMD ["httpd", "-D", "FOREGROUND"]
```

2. Erstellen Sie das benutzerdefinierte Apache-Image mit dem Namen do180/custom-apache.
 - 2.1. Überprüfen Sie das Containerfile für das benutzerdefinierte Apache-Image. Das Containerfile für das benutzerdefinierte Apache-Image sollte folgendermaßen aussehen:

```
FROM ubi8/ubi:8.3

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. Führen Sie den Befehl podman build aus, um das benutzerdefinierte Apache-Image zu erstellen, und nennen Sie es do180/custom-apache.

```
[student@workstation dockerfile-review]$ podman build --layers=false \
> -t do180/custom-apache .
STEP 1: FROM ubi8/ubi:8.3
...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...
```

Kapitel 5 | Erstellen benutzerdefinierter Container-Images

```
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/
STEP 9: CMD ["httpd", "-D", "FOREGROUND"]
STEP 10: COMMIT ...output omitted... localhost/do180/custom-apache:latest
...output omitted...
```

- 2.3. Führen Sie den Befehl `podman images` aus, um zu überprüfen, ob das benutzerdefinierte Image erstellt wurde.

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. Erstellen Sie einen neuen Container im getrennten Modus mit den folgenden Merkmalen:

- Name: `containerfile`
- Container-Image: `do180/custom-apache`
- Portweiterleitung: von Host-Port 20080 zu Container-Port 8080
- Als Daemon ausführen: ja

Überprüfen Sie, ob der Container bereit ist und ausgeführt wird.

- 3.1. Erstellen Sie den Container und führen Sie ihn aus.

```
[student@workstation dockerfile-review]$ podman run -d \
> --name containerfile -p 20080:8080 do180/custom-apache
367823e35c4a...
```

- 3.2. Überprüfen Sie, ob der Container bereit ist und ausgeführt wird.

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

4. Verifizieren Sie, dass der Server die HTML-Datei bereitstellt.

- 4.1. Führen Sie einen `curl`-Befehl auf `127.0.0.1:20080` aus.

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

Die Ausgabe sollte wie folgt aussehen:

```
<html>
<header><title>D0180 Hello!</title></header>
<body>
  Hello World! The containerfile-review lab works!
</body>
</html>
```

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem **workstation**-Rechner den Befehl **lab dockerfile-review grade** ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf **workstation** den Befehl **lab dockerfile-review finish** aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Ein **Containerfile** enthält Anweisungen, die angeben, wie ein Container-Image erstellt wird.
- Aus dem Red Hat Container Catalog oder von Quay.io bereitgestellte Container-Images eignen sich als Startpunkt für die Erstellung benutzerdefinierter Images für eine spezifische Sprache bzw. Technologie.
- Der Erstellungsprozess für Images aus einem Containerfile besteht aus den folgenden drei Schritten:
 1. Erstellen eines Arbeitsverzeichnisses.
 2. Angeben der Build-Anweisungen in einer **Containerfile**-Datei.
 3. Erstellen des Images mit dem Befehl `podman build`.
- Der Source-to-Image (S2I)-Prozess bietet eine Alternative zu Containerfiles. S2I implementiert einen standardisierten Container-Image-Build-Prozess für gängige Technologien anhand des Quellcodes der Anwendung. Dadurch können sich Entwickler auf die Anwendungsentwicklung konzentrieren und nicht auf die Containerfile-Entwicklung.

Kapitel 6

Bereitstellen containerisierter Anwendungen in OpenShift

Ziel

Bereitstellen von Anwendungen in einem einzelnen Container in OpenShift Container Platform

Ziele

- Beschreiben der Architektur von Kubernetes und von Red Hat OpenShift Container Platform.
- Erstellen von Kubernetes-Standardressourcen.
- Erstellen einer Route zu einem Service
- Erstellen einer Anwendung mit der Source-to-Image-Funktion in OpenShift Container Platform
- Erstellen einer Anwendung mit der OpenShift-Webkonsole

Abschnitte

- Beschreiben der Kubernetes- und OpenShift-Architektur (und Test)
- Erstellen von Kubernetes-Ressourcen (und angeleitete Übung)
- Erstellen von Weiterleitungen (und angeleitete Übung)
- Erstellen von Anwendungen mit der Source-to-Image-Funktion (und angeleitete Übung)
- Erstellung von Anwendungen mit der OpenShift Web Console (und angeleitete Übung)

Praktische Übung

- Bereitstellen containerisierter Anwendungen in OpenShift

Beschreiben der Kubernetes- und OpenShift-Architektur

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Beschreibung der Architektur eines Kubernetes-Clusters, der auf der Red Hat OpenShift Container Platform (RHOC) ausgeführt wird.
- Aufzählung der wichtigsten Ressourcentypen, die von Kubernetes und RHOC bereitgestellt werden.
- Identifizierung der Netzwerkmerkmale von Containern, Kubernetes und RHOC.
- Aufzählung der Mechanismen zur externen Bereitstellung eines Pods.

Kubernetes und OpenShift

In den vorherigen Kapiteln wurde beschrieben, dass Kubernetes ein Orchestrierungsservice ist, der die Bereitstellung, Verwaltung und Skalierung von Container-Anwendungen vereinfacht. Ein Hauptvorteil der Verwendung von Kubernetes besteht darin, dass mehrere Knoten verwendet werden, um die Ausfallsicherheit und Skalierbarkeit der verwalteten Anwendungen sicherzustellen. Kubernetes bildet einen Cluster von Knotenservern, die in Containern ausgeführt und zentral von einer Reihe von Control Plane-Servern verwaltet werden. Ein Server kann als Server und als Knoten fungieren, aber diese Rollen sind aus Gründen einer höheren Stabilität normalerweise voneinander getrennt.

Kubernetes Terminologie

Begriff	Definition
Knoten	Ein Server, der Anwendungen in einem Kubernetes-Cluster hostet.
Kontrollschicht	Stellt grundlegende Cluster-Services wie APIs oder Controller bereit.
Server-Knoten	Dieser Knoten führt Workloads für den Cluster aus. Anwendungs-Pods werden auf Server-Knoten geplant.
Ressource	Ressourcen sind jegliche Art von Komponentendefinitionen, die von Kubernetes verwaltet werden. Ressourcen enthalten die Konfiguration der verwalteten Komponente (beispielsweise die einem Knoten zugewiesene Rolle) und den aktuellen Status der Komponente (beispielsweise, wenn der Knoten verfügbar ist).
Controller	Ein Controller ist ein Kubernetes-Prozess, der Ressourcen überwacht und Änderungen vornimmt, um den aktuellen Status in den gewünschten Status zu verschieben.
Label	Ein Schlüssel-Wert-Paar, das einer beliebigen Kubernetes-Ressource zugeordnet werden kann. Selektoren verwenden Bezeichnungen zum Filtern geeigneter Ressourcen für die Planung und andere Vorgänge.

Begriff	Definition
Namespace	Ein Bereich für Kubernetes-Ressourcen und -Prozesse, sodass Ressourcen mit demselben Namen in unterschiedlichen Grenzen verwendet werden können.

**Anmerkung**

Die neuesten Versionen von Kubernetes implementieren viele Controller als Operatoren. Operatoren sind Kubernetes-Plug-in-Komponenten, die auf Clusterereignisse reagieren und den Status von Ressourcen steuern können. Operatoren und CoreOS Operator Framework werden in diesem Dokument nicht beschrieben.

Red Hat OpenShift Container Platform ist ein Satz modularer Komponenten und Services, die auf Red Hat CoreOS und Kubernetes basieren. RHOPC fügt PaaS-Funktionen wie Remote-Management, erhöhte Sicherheit, Lifecycle-Management für Anwendungen und Self-Service-Schnittstellen für Entwickler hinzu.

Ein OpenShift-Cluster ist ein Kubernetes-Cluster, der auf die gleiche Weise verwaltet werden kann, jedoch unter Verwendung der OpenShift-Managementtools, z. B. der Befehlszeilenschnittstelle oder der Web Console. Dies ermöglicht produktivere Arbeitsabläufe und erleichtert häufige Aufgaben erheblich.

OpenShift-Terminologie

Begriff	Definition
Infra-Knoten	Ein Knotenserver, der InfrastrukturServices wie Überwachung, Protokollierung oder externes Routing enthält.
Konsole	Eine vom RHOPC-Cluster bereitgestellte Web-Benutzeroberfläche, über die Entwickler und Administratoren mit Clusterressourcen interagieren können.
Projekt	Erweiterung von OpenShift der Namespaces von Kubernetes. Ermöglicht die Definition der Benutzerzugriffssteuerung (UAC) auf Ressourcen.

Im folgenden Schema wird der OpenShift Container Platform-Stack veranschaulicht:

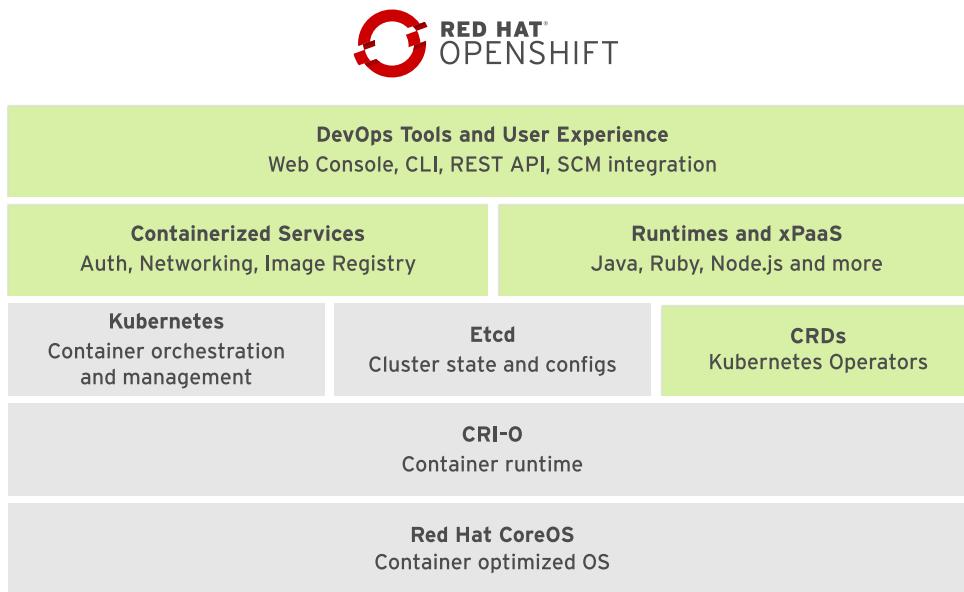


Abbildung 6.1: OpenShift-Komponenten-Stack

Die grundlegende durch Red Hat integrierte und erweiterte Container-Infrastruktur ist von unten nach oben und von links nach rechts dargestellt:

- Das Basisbetriebssystem ist Red Hat CoreOS. Red Hat CoreOS ist eine Linux-Distribution, die auf die Bereitstellung eines unveränderlichen Betriebssystems für die Container-Ausführung ausgerichtet ist.
- CRI-O ist eine Implementierung des Kubernetes Container Runtime Interface (CRI) für die Verwendung von mit Open Container Initiative (OCI) kompatiblen Laufzeiten. CRI-O kann jede Container-Laufzeit verwenden, die CRI erfüllt: runc (wird vom Docker-Service verwendet), libpod (wird von Podman verwendet) oder rkt (von CoreOS).
- Kubernetes verwaltet einen Cluster mit Host, physische oder virtuelle, die Container ausführen. Es verwendet Ressourcen, die Multicontainer-Anwendungen beschreiben, die aus mehreren Ressourcen bestehen und miteinander kommunizieren.
- Etcld ist ein verteilter Schlüssel-/Wert-Speicher, der von Kubernetes zum Speichern der Konfigurations- und Statusinformationen der Container und anderer Ressourcen im Kubernetes-Cluster verwendet wird.
- Benutzerdefinierte Ressourcendefinitionen (CRDs) sind Ressourcentypen, die in Etcld gespeichert und von Kubernetes verwaltet werden. Diese Ressourcentypen bilden den Status und die Konfiguration aller von OpenShift verwalteten Ressourcen.
- Containerisierte Services nehmen viele PaaS-Infrastrukturfunktionen wahr, wie Netzwerk und Autorisierung. RHOPC verwendet die grundlegende Container-Infrastruktur von Kubernetes und die zugrunde liegende Container-Laufzeit für die meisten internen Funktionen. Das heißt, die meisten internen RHOPC-Services werden als von Kubernetes orchestrierte Container ausgeführt.
- Laufzeiten und xPaaS sind grundlegende, gebrauchsfertige Container-Images für Entwickler, jedes ist mit einer bestimmten Laufzeitsprache oder einer Datenbank vorkonfiguriert. xPaaS stellt eine Reihe von grundlegenden Images für Red Hat-Middleware-Produkte, wie JBoss EAP und ActiveMQ, bereit. Red Hat OpenShift Application Runtimes (RHOR) sind eine Reihe von Laufzeiten, die für native Cloud-Anwendungen in OpenShift optimiert sind. Die verfügbaren

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

Anwendungslaufzeiten sind Red Hat JBoss EAP, OpenJDK, Thorntail, Eclipse Vert.x, Spring Boot und Node.js.

- RHOCP bietet Managementtools für die Web-UI und die CLI zur Verwaltung von Benutzeranwendungen und RHOCP-Services. Die Web-UI- und CLI-Tools von OpenShift sind aus REST-APIs erstellt, die von externen Tools wie IDEs und CI-Plattformen verwendet werden können.

Diese Illustration der OpenShift- und Kubernetes-Architektur bietet einen genaueren Einblick in die Zusammenarbeit der Infrastrukturkomponenten.

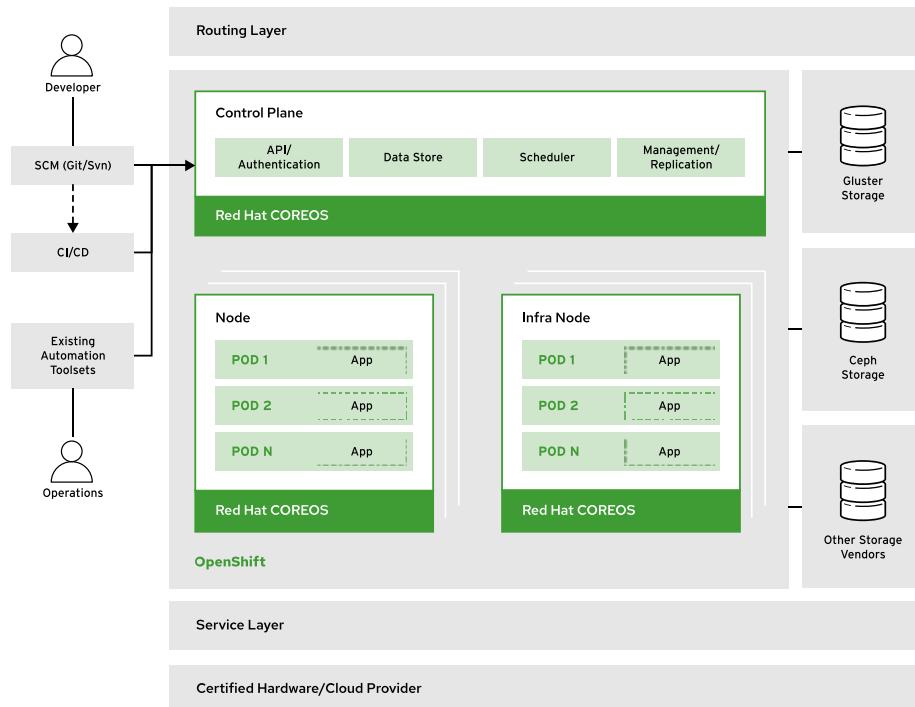


Abbildung 6.2: OpenShift- und Kubernetes-Architektur

Neue Funktionen in RHOCP 4

RHOCP 4 ist eine tiefgreifende Änderung gegenüber früheren Versionen. Neben der Abwärtskompatibilität mit früheren Versionen enthält es neue Funktionen, wie:

- CoreOS als obligatorisches Betriebssystem für alle Knoten mit einer unveränderlichen Infrastruktur, die für Container optimiert ist.
- Ein brandneues Cluster-Installationsprogramm, das den Installations- und Aktualisierungsprozess steuert.
- Eine Selbstverwaltungsplattform, die Cluster-Updates und -Wiederherstellungen ohne Unterbrechung automatisch durchführen kann.
- Ein neu konzipiertes Lifecycle-Management von Anwendungen.
- Ein Operator-SDK zum Erstellen, Testen und Packen von Operatoren.

Beschreiben von Kubernetes-Ressourcentypen

Kubernetes bietet sechs wichtige Ressourcentypen, die mithilfe einer YAML- bzw. JSON-Datei oder mithilfe der Managementtools von OpenShift erstellt und konfiguriert werden können:

Pods (po)

Repräsentieren eine Sammlung von Containern, die dieselben Ressourcen verwenden, zum Beispiel IP-Adressen und Volumes für persistenten Storage. Dies ist die zugrunde liegende Arbeitseinheit für Kubernetes.

Services (SVC)

Definieren eine einzelne IP-/Port-Kombination, die den Zugriff auf einen Pool von Pods ermöglicht. Services stellen Verbindungen zwischen Clients und Pods in der Regel mit dem Roundrobin-Verfahren her.

Replikationscontroller (RC)

Eine Kubernetes-Ressource, die definiert, wie Pods auf unterschiedliche Knoten repliziert werden (horizontal skaliert). Replikationscontroller sind ein grundlegender Kubernetes-Service, der eine hohe Verfügbarkeit für Pods und Container bietet.

Persistente Volumes (PV)

Definieren Speicherbereiche, die von Kubernetes-Pods verwendet werden sollen.

Anforderungen für ein persistentes Volume (PVC)

Bilden eine Storage-Anforderung durch einen Pod nach. PVCs verbinden ein PV mit einem Pod, sodass die Container ihn verwenden können, üblicherweise durch Einhängen des Storages in das Dateisystem des Containers.

ConfigMaps (cm) und Geheimnisse

Enthält eine Reihe von Schlüsseln und Werten, die von anderen Ressourcen verwendet werden können. ConfigMaps und Geheimnisse werden normalerweise verwendet, um Konfigurationswerte zu zentralisieren, die von mehreren Ressourcen verwendet werden. Geheimnisse unterscheiden sich von ConfigMaps-Zuordnungen dadurch, dass die Werte von Geheimnissen immer codiert werden (nicht verschlüsselt) und deren Zugriff auf weniger autorisierte Benutzer beschränkt ist.



Anmerkung

Auch wenn Kubernetes-Pods eigenständig erstellt werden können, werden sie in der Regel von übergeordneten Ressourcen wie Replikationscontrollern erstellt.

OpenShift-Ressourcentypen

Im Folgenden sehen Sie die wichtigsten Ressourcentypen, die von OpenShift Container Platform zu Kubernetes hinzugefügt wurden:

Deployment und DeploymentConfig (DC)

In OpenShift 4.5 wurde das Konzept der Deployment-Ressourcen eingeführt, um DeploymentConfig als Standardkonfiguration für Pods zu ersetzen. Beide stellen einen Satz von Containern dar, die in einem Pod enthalten sind, sowie die zu verwendenden Bereitstellungsstrategien. Enthalten ist die Konfiguration, die auf alle Container jedes Pod-Replikats angewendet werden soll, z. B. das Basis-Image, Tags, Storage-Definitionen und die Befehle, die ausgeführt werden sollen, wenn die Container gestartet werden.

Das Objekt Deployment dient als verbesserte Version des Objekts DeploymentConfig. Im Folgenden finden Sie einige Ersetzungen von Funktionen für die beiden Objekte:

- Das automatische Rollback wird nicht mehr von Bereitstellungsobjekten unterstützt.
- Jede Änderung in der Pod-Vorlage, die von Bereitstellungsobjekten verwendet wird, löst automatisch ein neues Rollout aus.
- Lifecycle-Hooks werden nicht mehr von Bereitstellungsobjekten unterstützt.
- Der Bereitstellungsprozess eines Bereitstellungsobjekts kann jederzeit angehalten werden, ohne dass dies Auswirkungen auf den Deployer-Prozess hat.
- Ein Bereitstellungsobjekt kann über so viele aktive Replikatsätze verfügen, wie der Benutzer möchte, und alte Replikate werden nach einer Weile herunterskaliert. Im Gegensatz dazu kann das Objekt „deploymentconfig“ jeweils nur zwei aktive Replikationssätze aufweisen.

Selbst wenn Deployment-Objekte als standardmäßiger Ersatz für DeploymentConfig-Objekte fungieren sollen, können Benutzer in OpenShift 4.6 diese weiterhin verwenden, wenn sie eine spezielle Funktion benötigen, die von diesen Objekten bereitgestellt wird. In diesem Fall ist es erforderlich, den Objekttyp beim Erstellen einer neuen Anwendung durch Angabe des Flags `-as-deployment-config` anzugeben.

Build-Konfiguration (BC)

Definiert einen Prozess, der im OpenShift-Projekt ausgeführt werden soll. Werden von der Source-to-Image-Funktion (S2I) von OpenShift zur Erstellung eines Container-Images über einen Anwendungsquellcode verwendet, der in einem Git-Repository gespeichert ist. bcs stellen zusammen mit dcs einen grundlegenden, aber erweiterbaren Continuous Integration und Continuous Delivery-Workflow zur Verfügung.

Routen

Repräsentieren einen DNS-Hostnamen, der vom OpenShift-Router als Eingangspunkt für Anwendungen und Mikroservices erkannt wird.



Anmerkung

Führen Sie den Befehl `oc api-resources` oder `kubectl api-resources` aus, um eine Liste aller in einem RHOP-Cluster verfügbaren Ressourcen und deren Abkürzungen zu erhalten.

Auch wenn Kubernetes-Replikationscontroller in OpenShift eigenständig erstellt werden können, werden sie in der Regel von Ressourcen einer höheren Stufe wie Bereitstellungscontrollern erstellt.

Netzwerke

Jeder in einem Kubernetes-Cluster bereitgestellte Container verfügt über eine IP-Adresse, die über ein internes Netzwerk zugewiesen wird, und auf die nur über den Knoten zugegriffen werden kann, auf dem der Container ausgeführt wird. Aufgrund der Kurzlebigkeit von Containern werden IP-Adressen kontinuierlich zugewiesen und freigegeben.

Kubernetes bietet ein softwaredefiniertes Netzwerk (SDN), das sich über die internen Container-Netzwerke mehrerer Knoten erstreckt und Containern ermöglicht, über einen beliebigen Pod innerhalb eines Hosts auf Pods anderer Hosts zuzugreifen. Der Zugriff auf SDN ist nur innerhalb desselben Kubernetes-Clusters möglich.

Container in Kubernetes-Pods sollten untereinander keine direkte Verbindung zur dynamischen IP-Adresse herstellen. Services lösen dieses Problem, indem stabilere IP-Adressen vom SDN mit den Pods verbunden werden. Wenn Pods neu gestartet, repliziert oder auf verschiedenen Knoten neu geplant werden, werden die Services aktualisiert und bieten Skalierbarkeit und Fehlertoleranz.

Der externe Zugriff auf Container ist komplizierter. Kubernetes-Services können ein **NodePort**-Attribut angeben. Dieses stellt einen Netzwerkport dar, der von allen Cluster-Knoten zum SDN umgeleitet wird. Dann können die Container im Knoten einen Port an den Port des Knotens umleiten. Die Skalierbarkeit ist leider bei all diesen Ansätzen unzureichend.

OpenShift macht den externen Zugriff auf Container sowohl skalierbar als auch einfacher, indem Routenressourcen definiert werden. Eine Route definiert extern erreichbare DNS-Namen und -Ports für einen Service. Ein Router (Ingress-Controller) leitet HTTP- und TLS-Anforderungen an die Serviceadressen im Kubernetes-SDN weiter. Es müssen lediglich die gewünschten DNS-Namen den IP-Adressen der Router-Knoten von RHOCP zugeordnet werden.



Literaturhinweise

Website zur Kubernetes-Dokumentation

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

Website zur OpenShift-Dokumentation

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

Understanding Operators

<https://docs.openshift.com/container-platform/4.6/operators/olm-what-operators-are.html>

► Quiz

Beschreibung von Kubernetes und OpenShift

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

► 1. Welche zwei Aussagen treffen im Hinblick auf die Kubernetes-Architektur zu? (Wählen Sie zwei Antworten aus.)

- a. Kubernetes-Knoten können ohne Control Plane verwaltet werden.
- b. Kubernetes-Control Planes verwalten die Skalierung von Pods.
- c. Kubernetes-Control Planes planen Pods auf spezifischen Knoten.
- d. Kubernetes-Tools können nicht zur Verwaltung von Ressourcen in einem OpenShift-Cluster verwendet werden.
- e. Container, die über Kubernetes-Pods erstellt werden, können nicht mithilfe von Tools, beispielsweise Podman, verwaltet werden.

► 2. Welche beiden Aussagen treffen im Hinblick auf Kubernetes- und OpenShift-Ressourcen zu? (Wählen Sie zwei Antworten aus.)

- a. Pods sind für die Bereitstellung des eigenen persistenten Storage zuständig.
- b. Alle Pods, die über denselben Replikationscontroller erstellt werden, müssen auf demselben Knoten ausgeführt werden.
- c. Ein persistentes Volume wird von den Pods verwendet, um Storage-Bereiche für die persistenten Daten zuzuweisen.
- d. Ein Replikationscontroller ist für die Überwachung und Verwaltung der Anzahl der Pods für eine bestimmte Anwendung verantwortlich.

► 3. Welche beiden Aussagen treffen im Hinblick auf Kubernetes- und OpenShift-Netzwerke zu? (Wählen Sie zwei Antworten aus.)

- a. Kubernetes-Services können eine IP-Adresse für den Zugriff auf eine Gruppe von Pods bereitstellen.
- b. Kubernetes ist für die Bereitstellung eines vollständig qualifizierten Domänenamens für einen Pod zuständig.
- c. Replikationscontroller sind für das Routing externer Anforderungen an den Pod zuständig.
- d. Routen sind für die Bereitstellung von DNS-Namen für den externen Zugriff zuständig.

► **4. Welche Aussage trifft im Hinblick auf persistenten Storage in OpenShift und Kubernetes zu?**

- a. Eine Anforderung für persistenten Storage repräsentiert einen Storage-Bereich, den Pods zur Speicherung von Daten verwenden können. Dieser wird vom Anwendungsentwickler bereitgestellt.
- b. Eine Anforderung für persistenten Storage stellte einen Storage-Bereich dar, der von einem Pod für die Speicherung von Daten angefordert werden kann. Dieser wird jedoch vom Cluster-Administrator bereitgestellt.
- c. Eine Anforderung für persistenten Storage ist die Speichermenge, die einem Knoten zugewiesen werden kann, damit Entwickler ermitteln können, wie viel Speicher sie für die Ausführung ihrer Anwendung benötigen.
- d. Eine Anforderung für persistenten Storage ist die Anzahl der CPU-Verarbeitungseinheiten, die einem Anwendungs-Pod zugewiesen werden können. Diese unterliegen einer vom Cluster-Administrator festgelegten Beschränkung.

► **5. Welche Aussage trifft im Hinblick auf OpenShift-Ergänzungen zu Kubernetes zu?**

- a. OpenShift fügt Funktionen hinzu, um die Kubernetes-Konfiguration vieler praktischer Anwendungsfälle zu vereinfachen.
- b. Von OpenShift erstellte Container-Images können nicht allein mit Kubernetes verwendet werden.
- c. Red Hat verwaltet im RHOCUP-Produkt gegabelte Versionen von Kubernetes.
- d. Für die Verwendung von Continuous Integration und Continuous Deployment in Kombination mit RHOCUP werden externe Tools benötigt.

► Lösung

Beschreibung von Kubernetes und OpenShift

Wählen Sie die richtigen Antworten auf die folgenden Fragen aus:

► 1. Welche zwei Aussagen treffen im Hinblick auf die Kubernetes-Architektur zu? (Wählen Sie zwei Antworten aus.)

- a. Kubernetes-Knoten können ohne Control Plane verwaltet werden.
- b. Kubernetes-Control Planes verwalten die Skalierung von Pods.
- c. Kubernetes-Control Planes planen Pods auf spezifischen Knoten.
- d. Kubernetes-Tools können nicht zur Verwaltung von Ressourcen in einem OpenShift-Cluster verwendet werden.
- e. Container, die über Kubernetes-Pods erstellt werden, können nicht mithilfe von Tools, beispielsweise Podman, verwaltet werden.

► 2. Welche beiden Aussagen treffen im Hinblick auf Kubernetes- und OpenShift-Ressourcen zu? (Wählen Sie zwei Antworten aus.)

- a. Pods sind für die Bereitstellung des eigenen persistenten Storage zuständig.
- b. Alle Pods, die über denselben Replikationscontroller erstellt werden, müssen auf demselben Knoten ausgeführt werden.
- c. Ein persistentes Volume wird von den Pods verwendet, um Storage-Bereiche für die persistenten Daten zuzuweisen.
- d. Ein Replikationscontroller ist für die Überwachung und Verwaltung der Anzahl der Pods für eine bestimmte Anwendung verantwortlich.

► 3. Welche beiden Aussagen treffen im Hinblick auf Kubernetes- und OpenShift-Netzwerke zu? (Wählen Sie zwei Antworten aus.)

- a. Kubernetes-Services können eine IP-Adresse für den Zugriff auf eine Gruppe von Pods bereitstellen.
- b. Kubernetes ist für die Bereitstellung eines vollständig qualifizierten Domänenamens für einen Pod zuständig.
- c. Replikationscontroller sind für das Routing externer Anforderungen an den Pod zuständig.
- d. Routen sind für die Bereitstellung von DNS-Namen für den externen Zugriff zuständig.

► **4. Welche Aussage trifft im Hinblick auf persistenten Storage in OpenShift und Kubernetes zu?**

- a. Eine Anforderung für persistenten Storage repräsentiert einen Storage-Bereich, den Pods zur Speicherung von Daten verwenden können. Dieser wird vom Anwendungsentwickler bereitgestellt.
- b. Eine Anforderung für persistenten Storage stellte einen Storage-Bereich dar, der von einem Pod für die Speicherung von Daten angefordert werden kann. Dieser wird jedoch vom Cluster-Administrator bereitgestellt.
- c. Eine Anforderung für persistenten Storage ist die Speichermenge, die einem Knoten zugewiesen werden kann, damit Entwickler ermitteln können, wie viel Speicher sie für die Ausführung ihrer Anwendung benötigen.
- d. Eine Anforderung für persistenten Storage ist die Anzahl der CPU-Verarbeitungseinheiten, die einem Anwendungs-Pod zugewiesen werden können. Diese unterliegen einer vom Cluster-Administrator festgelegten Beschränkung.

► **5. Welche Aussage trifft im Hinblick auf OpenShift-Ergänzungen zu Kubernetes zu?**

- a. OpenShift fügt Funktionen hinzu, um die Kubernetes-Konfiguration vieler praktischer Anwendungsfälle zu vereinfachen.
- b. Von OpenShift erstellte Container-Images können nicht allein mit Kubernetes verwendet werden.
- c. Red Hat verwaltet im RHOCUP-Produkt gegabelte Versionen von Kubernetes.
- d. Für die Verwendung von Continuous Integration und Continuous Deployment in Kombination mit RHOCUP werden externe Tools benötigt.

Erstellen von Kubernetes-Ressourcen

Ziele

Am Ende dieses Abschnitts sollten Teilnehmer in der Lage sein, Kubernetes-Standardressourcen zu erstellen.

Das Red Hat OpenShift Container Platform-Befehlszeilentool (RHOCP)

Die Hauptmethode für die Interaktion mit einem RHOCP-Cluster ist die Verwendung des Befehls `oc`. Die grundlegende Verwendung des Befehls erfolgt durch seine Unterbefehle in der folgenden Syntax:

```
$> oc <command>
```

Bevor Sie mit einem Cluster interagieren, erfordern die meisten Vorgänge, dass sich der Benutzer anmeldet. Die Syntax für die Anmeldung wird unten gezeigt:

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

Beschreiben der Syntax der Ressourcendefinition für Pods

RHOCP führt Container in Kubernetes-Pods aus. Zur Erstellung eines Pods über ein Container-Image benötigt OpenShift eine *Pod-Ressourcendefinition*. Diese kann als JSON- oder YAML-Textdatei bereitgestellt werden. Alternativ können Ressourcendefinitionen mithilfe des Befehls `oc new-app` über die Standardeinstellungen oder mithilfe der OpenShift-Webkonsole erstellt werden.

Ein Pod ist eine Sammlung von Containern und anderen Ressourcen. Im Folgenden sehen Sie ein Beispiel für die Pod-Definition eines WildFly-Anwendungsservers im YAML-Format:

```
apiVersion: v1
kind: Pod❶
metadata:
  name: wildfly❷
  labels:
    name: wildfly❸
spec:
  containers:
    - resources:
        limits :
          cpu: 0.5
      image: do276/todojee
      name: wildfly
      ports:
        - containerPort: 8080❹
```

```

        name: wildfly
      env: ⑤
        - name: MYSQL_ENV_MYSQL_DATABASE
          value: items
        - name: MYSQL_ENV_MYSQL_USER
          value: user1
        - name: MYSQL_ENV_MYSQL_PASSWORD
          value: mypa55
    
```

- ① Deklariert einen Kubernetes-Pod-Ressourcentyp.
- ② Ein eindeutiger Name für einen Pod in Kubernetes, der Administratoren ermöglicht, Befehle auf dem Pod auszuführen.
- ③ Erstellt eine Bezeichnung mit einem Schlüssel namens `name`, den andere Ressourcen in Kubernetes in der Regel als Service verwenden können, um ihn zu finden.
- ④ Ein containerabhängiges Attribut, das festlegt, welcher Port auf dem Container bereitgestellt wird.
- ⑤ Definiert eine Sammlung von Umgebungsvariablen.

Einige Pods erfordern unter Umständen Umgebungsvariablen, die von einem Container gelesen werden können. Kubernetes wandelt alle Name- und Wert-Paare in Umgebungsvariablen um. Beispiel: Die Variable `MYSQL_ENV_MYSQL_USER` wird intern über die Kubernetes-Laufzeit mit dem Wert `user1` deklariert und an die Container-Image-Definition weitergeleitet. Da der Container zum Abrufen der Anmeldeinformation des Benutzers denselben Variablennamen verwendet, wird der Wert der WildFly-Container-Instanz zur Bestimmung des Benutzernamens verwendet, über den auf eine MySQL-Datenbankinstanz zugegriffen wird.

Beschreiben der Syntax der Serviceressourcendefinition

Kubernetes bietet ein virtuelles Netzwerk, das Pods ermöglicht, sich über verschiedene Server-Knoten zu verbinden. Kubernetes bietet jedoch keine einfache Methode für Pods zur Ermittlung der IP-Adressen anderer Pods:

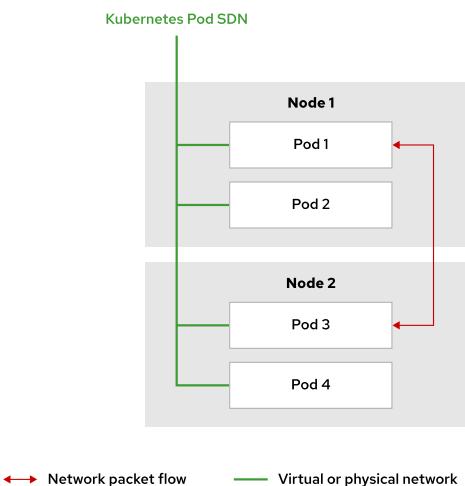


Abbildung 6.3: Grundlegendes Kubernetes-Networking

Wenn Pod 3 fehlschlägt und neu gestartet wird, kann er nach dem Neustart eine andere IP-Adresse aufweisen. Dies würde dazu führen, dass Pod 1 bei dem Versuch, mit Pod 3 zu kommunizieren, fehlschlägt. Eine Serviceschicht bietet die Abstraktion, die zum Lösen dieses Problems erforderlich ist.

Services sind essentielle Ressourcen für jede OpenShift-Anwendung. Sie ermöglichen es Containern in einem Pod, Netzwerkverbindungen zu Containern in einem anderen Pod zu öffnen. Ein Pod kann aus verschiedenen Gründen neu gestartet werden. Dabei erhält er jedes Mal eine andere interne IP-Adresse. Anstatt die IP-Adresse eines anderen Pods nach jedem Neustart ermitteln zu müssen, stellen Services eine stabile IP-Adresse bereit, unabhängig davon, auf welchem Server-Knoten der Pod nach dem Neustart ausgeführt wird:

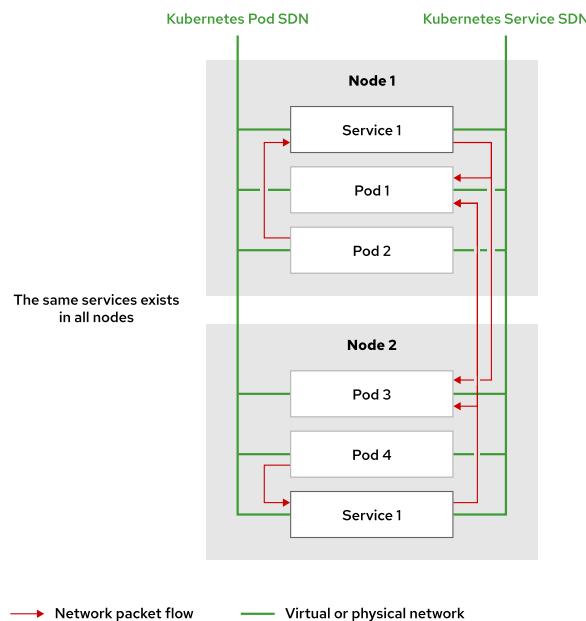


Abbildung 6.4: Kubernetes-Servicenetzwerke

Die meisten gängigen Anwendungen werden nicht als einzelner Pod ausgeführt. Sie müssen horizontal skaliert werden. Das bedeutet, dass viele Pods dieselben Container über dieselbe Pod-Ressourcendefinition ausführen, um die wachsende Nachfrage der Benutzer zufriedenzustellen. Ein Service ist an eine Reihe von Pods gebunden. Diesen Pods stellt der Service eine einzelne IP-Adresse und einen Client zum Lastenausgleich zwischen den Pods bereit.

Die Pod-Gruppe, die im Hintergrund eines Services ausgeführt wird, wird von einer Deployment-Ressource verwaltet. Über die Deployment-Ressource wird eine ReplicationController-Ressource integriert, die festlegt, wie viele Pod-Kopien (Replikationen) erstellt werden müssen. Zudem werden darüber neue Pods erstellt, wenn andere Pods ausfallen. Deployment- und ReplicationController-Ressourcen werden später in diesem Kapitel erläutert.

Im folgenden Beispiel sehen Sie die Minimaldefinition eines Services in der JSON-Syntax:

```
{
  "kind": "Service", ①
  "apiVersion": "v1",
  "metadata": {
    "name": "quotedb" ②
  },
}
```

```

    "spec": {
      "ports": [ ❸
        {
          "port": 3306,
          "targetPort": 3306
        }
      ],
      "selector": {
        "name": "mysqlDb" ❹
      }
    }
  }
}

```

- ❶ Art der Kubernetes-Ressource. In diesem Fall ein Service.
- ❷ Ein eindeutiger Name für den Service.
- ❸ ports ist ein Array von Objekten, das die Netzwerkports beschreibt, die von dem Service bereitgestellt werden. Das Attribut targetPort muss mit dem Attribut containerPort aus einer Pod-Container-Definition übereinstimmen, und das port-Attribut ist der vom Service bereitgestellte Port. Clients verbinden sich mit dem Serviceport und der Service leitet Pakete an den Pod targetPort weiter.
- ❹ selector bestimmt die Vorgehensweise des Services bei der Suche nach Pods, an die Pakete weitergeleitet werden. Die Ziel-Pods müssen übereinstimmende Bezeichnungen in ihren Metadaten-Attributen enthalten. Wenn der Service mehrere Pods mit übereinstimmenden Bezeichnungen findet, sorgt er für einen Lastenausgleich zwischen den betreffenden Netzwerkverbindungen.

Jedem Service wird eine eindeutige IP-Adresse zugewiesen, über die sich Clients verbinden können. Diese IP-Adresse stammt aus einem anderen internen OpenShift SDN, das sich vom internen Pod-Netzwerk unterscheidet, jedoch nur für Pods sichtbar ist. Jeder Pod, dem mit dem Selektor (selector) übereinstimmt, wird als Endpunkt zur Serviceressource hinzugefügt.

Ermittlung von Services

Eine Anwendung findet in der Regel eine Service-IP-Adresse und einen Port mithilfe von Umgebungsvariablen. Für jeden Service in einem OpenShift-Projekt werden die folgenden Umgebungsvariablen automatisch definiert und für alle Pods, die sich im selben Projekt befinden, in Container eingefügt:

- `SVC_NAME_SERVICE_HOST` ist die IP-Adresse des Service.
- `SVC_NAME_SERVICE_PORT` ist der TCP-Port des Service.



Anmerkung

Der `SVC_NAME`-Teil der Variablen wird den Einschränkungen bei der DNS-Namenszuweisung entsprechend geändert: Buchstaben werden großgeschrieben und Unterstriche (`_`) durch Bindestriche (`-`) ersetzt.

Sie können Services über einen Pod auch mithilfe des internen DNS-Servers von OpenShift ermitteln; dieser ist nur für Pods sichtbar. Jedem Service wird dynamisch ein SRV-Datensatz mit einem FQDN im folgenden Format zugewiesen:

```
SVC_NAME .PROJECT_NAME.svc.cluster.local
```

Bei der Ermittlung von Services mithilfe von Umgebungsvariablen dürfen Pods erst nach Erstellung des Services erstellt und gestartet werden. Wenn die Anwendung jedoch so konfiguriert wurde, dass Services mittels DNS-Abfragen ermittelt werden, können Services gesucht werden, die nach dem Starten des Pods erstellt wurden.

Für Anwendungen, die außerhalb eines OpenShift-Clusters auf den Service zugreifen, stehen zwei Methoden zur Auswahl:

1. **NodePort**: Dies ist ein alter Kubernetes-basierter Ansatz, bei dem der Service externen Clients bereitgestellt wird, indem er mit den auf dem Server-Knoten-Host verfügbaren Ports verbunden wird, der anschließend die Verbindungen über einen Proxy an die IP-Adresse des Services leitet. Verwenden Sie den Befehl `oc edit svc`, um die Serviceattribute zu bearbeiten, und geben Sie `NodePort` als Wert für `type` und einen Port-Wert für das `nodePort`-Attribut an. OpenShift leitet dann Verbindungen unter Verwendung der öffentlichen IP-Adresse des Server-Knoten-Hosts und des in `NodePort` festgelegten Werts über einen Proxy an den Service weiter.
2. **OpenShift-Routen**: Dies ist in OpenShift der bevorzugte Ansatz zur Bereitstellung von Services unter Verwendung einer eindeutigen URL. Verwenden Sie den Befehl `oc expose`, um einen Service für den externen Zugriff oder einen Service über die Web-Konsole von OpenShift bereitzustellen.

Abbildung 6.5 veranschaulicht, wie NodePort-Services den externen Zugriff auf Kubernetes-Services ermöglichen. OpenShift-Routen werden im weiteren Verlauf dieses Kurses ausführlicher behandelt.

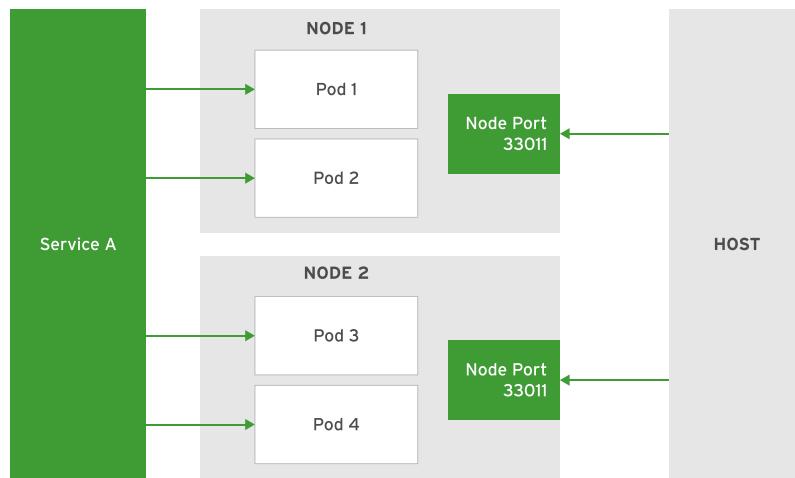


Abbildung 6.5: Alternative Methode für den externen Zugriff auf einen Kubernetes-Service

OpenShift stellt den Befehl `oc port-forward` bereit, um einen lokalen Port zu einem Pod-Port weiterzuleiten. Dies unterscheidet sich vom Zugriff auf einen Pod über eine Serviceressource:

- Die Zuordnung der Portweiterleitung wird nur auf der Workstation vorgenommen, in welcher der `oc`-Client ausgeführt wird, wohingegen ein Service einen Port für alle Netzwerkbenutzer zuweist.
- Während Services das Load Balancing für Verbindungen zu (potenziell) mehreren Pods vornehmen, werden bei einer Zuordnung von Portweiterleitungen Verbindungen an einen einzelnen Pod weitergeleitet.



Anmerkung

Red Hat rät von der Verwendung des NodePort-Ansatzes ab, um zu vermeiden, dass der Service direkten Verbindungen ausgesetzt wird. Die Zuordnung über Portweiterleitung in OpenShift gilt als sicherere Alternative.

Im folgenden Beispiel wird die Verwendung des `oc port-forward`-Befehls veranschaulicht:

```
[user@host ~]$ oc port-forward mysql-openshift-1-g1qrp 3306:3306
```

Der Befehl leitet Port 3306 vom Entwicklerrechner an Port 3306 im db-Pod weiter, wo ein MySQL-Server (in einem Container) Netzwerkverbindungen akzeptiert.



Anmerkung

Stellen Sie sicher, dass das Terminalfenster ausgeführt wird, wenn Sie diesen Befehl ausführen. Das Schließen des Fensters oder das Abbrechen des Prozesses beendet das Port-Mapping.

Erstellen neuer Anwendungen

Einfache Anwendungen, komplexe mehrstufige Anwendungen und Mikroservice-Anwendungen können mithilfe einer einzigen Ressourcendefinitionsdatei beschrieben werden. Diese Datei enthält viele Pod-Definitionen, Service-Definitionen für die Verbindung der Pods, Replikationscontroller oder Deployment-Ressourcen für die horizontale Skalierung der Anwendungs-Pods, PersistentVolumeClaims, um Anwendungsdaten beizubehalten, und alles andere, was von OpenShift verwaltet werden kann.

Mit dem Befehl `oc new-app` sowie der Option `-o json` oder `-o yaml` kann eine Definitionsdatei für eine Skeleton-Ressource im JSON- oder YAML-Format erstellt werden. Diese Datei kann mit dem Befehl `oc create -f <filename>` angepasst und für die Erstellung einer Anwendung verwendet werden, oder sie kann mit anderen Ressourcendefinitionsdateien zu einer zusammengesetzten Anwendung zusammengeführt werden.

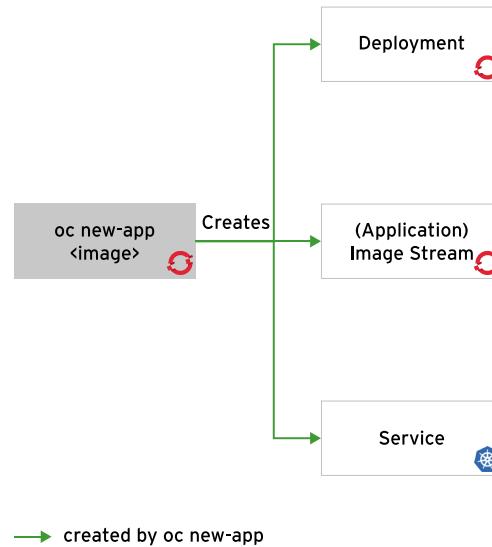
Mit dem Befehl `oc new-app` können Anwendungspods erstellt werden, die auf OpenShift auf unterschiedliche Art und Weise ausgeführt werden können. Mit diesem Befehl lassen sich anhand des Source-to-Image Prozesses (S2I-Prozess) Pods aus bereits bestehenden Docker-Images und aus Docker-Dateien oder Quellcode erstellen.

Führen Sie den Befehl `oc new-app -h` aus, um die verschiedenen zur Verfügung stehenden Optionen für die Erstellung neuer Anwendungen auf OpenShift nachzuvollziehen.

Der folgende Befehl erstellt eine Anwendung basierend auf einem Image, `mysql`, von Docker Hub, mit der Bezeichnung `db=mysql`:

```
[user@host ~]$ oc new-app mysql MYSQL_USER=user MYSQL_PASSWORD=pass  
MYSQL_DATABASE=testdb -l db=mysql
```

In der folgenden Abbildung werden die Kubernetes- und OpenShift-Ressourcen angezeigt, die durch den Befehl `oc new-app` erstellt werden, wenn das Argument ein Container-Image ist:

**Abbildung 6.6: Für eine neue Anwendung erstellte Ressourcen**

Der folgende Befehl erstellt eine Anwendung basierend auf einem Image aus einer privaten Docker Image-Registry:

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

Der folgende Befehl erstellt eine Anwendung auf der Grundlage von in einem Git-Repository gespeichertem Quellcode:

```
oc new-app https://github.com/openshift/ruby-hello-world --name=ruby-hello
```

Im nächsten Abschnitt erfahren Sie mehr über den S2I-Prozess (Source-to-Image) und dessen Konzepte und lernen weitere Methoden für die Verwendung des Befehls `oc new-app` zur Erstellung von Anwendungen für OpenShift kennen.

Mit dem folgenden Befehl wird eine Anwendung basierend auf einer vorhandenen Vorlage erstellt:

```
$ 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...
```



Anmerkung

Weitere Informationen zu Vorlagen finden Sie im nächsten Kapitel.

Verwalten von persistentem Storage

Zusätzlich zur Spezifikation von benutzerdefinierten Images können Sie persistenten Storage erstellen und an Ihre Anwendung anhängen. Auf diese Weise können Sie sicherstellen, dass Ihre Daten beim Löschen Ihrer Pods nicht verloren gehen. Führen Sie den Befehl `oc get pv` aus, um die PersistentVolume-Objekte in einem Cluster aufzulisten:

```
[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...
```

Führen Sie den Befehl `oc get` mit der Option `-o yaml` aus, um die YAML-Definition für ein bestimmtes PersistentVolume anzuseigen:

```
[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
```

Führen Sie den Befehl `oc create` aus, um einem Cluster weitere PersistentVolume-Objekte hinzuzufügen:

```
[admin@host ~]$ oc create -f pv1001.yaml
```



Anmerkung

Die obige Datei `pv1001.yaml` muss eine Definition des persistenten Volumes enthalten, deren Struktur der Ausgabe des Befehls `oc get pv pv-name -o yaml` ähnelt.

Anfordern von persistenten Volumes

Wenn eine Anwendung Storage benötigt, erstellen Sie ein `PersistentVolumeClaim`-Objekt (PVC), um eine dedizierte Storage-Ressource aus dem Clusterpool anzufordern. Der folgende Inhalt aus einer Datei mit dem Namen `pvc.yaml` ist eine Beispieldefinition für eine PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myapp
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Die PVC definiert Storage-Anforderungen für die Anwendungen, beispielsweise die Kapazität oder der Durchsatz. Führen Sie den Befehl `oc create` aus, um die PVC zu erstellen:

```
[admin@host ~]$ oc create -f pvc.yaml
```

Nachdem Sie eine PVC erstellt haben, versucht OpenShift, eine verfügbare `PersistentVolume`-Ressource zu finden, welche die Anforderungen der PVC erfüllt. Wenn OpenShift eine Übereinstimmung findet, bindet es das „`PersistentVolume`“-Objekt an das „`PersistentVolumeClaim`“-Objekt. Führen Sie den Befehl `oc get pvc` aus, um die PVCs in einem Projekt aufzulisten:

```
[admin@host ~]$ oc get pvc
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
myapp    Bound    pv0001   1Gi        RWO          6s
```

Die Ausgabe zeigt an, ob ein persistentes Volume zusammen mit den Attributen der PVC (beispielsweise die Kapazität) an die PVC gebunden ist.

Um das permanente Volume in einem Anwendungs-Pod zu verwenden, definieren Sie eine Volume-Bereitstellung für einen Container, der auf das `PersistentVolumeClaim`-Objekt verweist. Die folgende Definition des Anwendungs-Pods verweist auf ein `PersistentVolumeClaim`-Objekt, um eine Volume-Bereitstellung für die Anwendung zu definieren:

```
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 Dieser Abschnitt deklariert, dass das pvol-Volume unter /var/www/html im Container-Dateisystem eingebunden wird.
- ② Dieser Abschnitt definiert das pvol-Volume.
- ③ Das pvol-Volume verweist auf die myapp-PVC. Wenn OpenShift ein verfügbares persistentes Volume der myapp-PVC zuordnet, verweist das pvol-Volume auf dieses zugeordnete Volumen.

Verwalten von OpenShift-Ressourcen an der Befehlszeile

Für die Verwaltung der OpenShift-Ressourcen stehen zahlreiche wesentliche Befehle zur Verfügung, wie unten beschrieben.

Führen Sie den Befehl `oc get` aus, um Informationen über Ressourcen im Cluster abzurufen. Allgemein zeigt dieser Befehl nur die wichtigsten Eigenschaften der Ressourcen an und keine detaillierteren Informationen.

Der Befehl `oc get RESOURCE_TYPE` zeigt eine Zusammenfassung aller Ressourcen des angegebenen Typs an. Das folgende Beispiel veranschaulicht die Ausgabe des Befehls `oc get pods`.

NAME	READY	STATUS	RESTARTS	AGE
nginx-1-5r583	1/1	Running	0	1h
myapp-1-l44m7	1/1	Running	0	1h

oc get all

Führen Sie den Befehl `oc get all` aus, um eine Zusammenfassung der wichtigsten Komponenten eines Clusters abzurufen. Dieser Befehl wiederholt sich durch die wichtigsten Ressourcentypen für das aktuelle Projekt und druckt eine Zusammenfassung ihrer Informationen aus:

NAME	DOCKER REPO	TAGS	UPDATED	
is/nginx	172.30.1.1:5000/basic-kubernetes/nginx	latest	About an hour ago	
<hr/>				
NAME	REVISION	DESIRED	CURRENT	TRIGGERED BY
dc/nginx	1	1	1	config,image(nginx:latest)
<hr/>				
NAME	DESIRED	CURRENT	READY	AGE
rc/nginx-1	1	1	1	1h
<hr/>				
NAME	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
svc/nginx	172.30.72.75	<none>	80/TCP, 443/TCP	1h

NAME	READY	STATUS	RESTARTS	AGE
po/nginx-1-ypp8t	1/1	Running	0	1h

oc describe

Wenn die von `oc get` bereitgestellten Zusammenfassungen unzureichend sind, führen Sie den Befehl `oc describe RESOURCE_TYPE RESOURCE_NAME` aus, um zusätzliche Informationen abzurufen. Im Gegensatz zum Befehl `oc get` wiederholt sich dieser nicht durch die verschiedenen Ressourcen nach Typ. Obwohl die wichtigsten Ressourcen beschrieben werden können, ist diese Funktion nicht für alle Ressourcen verfügbar. Im Folgenden finden Sie ein Beispiel für das Ergebnis der Beschreibung einer Pod-Ressource:

```
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

Mit dem Befehl `oc get RESOURCE_TYPE RESOURCE_NAME` kann eine Ressourcendefinition exportiert werden. Typische Anwendungsfälle sind die Erstellung eines Backups oder die Änderung einer Definition. Der Befehl `-o yaml` gibt die Objektdarstellung im YAML-Format aus, dies kann allerdings mithilfe einer `-o json`-Option in JSON geändert werden.

oc create

Mit diesem Befehl werden aus einer Ressourcendefinition Ressourcen erstellt. Er geht normalerweise mit dem Befehl `oc get RESOURCE_TYPE RESOURCE_NAME -o yaml` für das Bearbeiten von Definitionen einher.

oc edit

Dieser Befehl ermöglicht Benutzern, Ressourcen aus einer Ressourcendefinition zu bearbeiten. Dieser Befehl öffnet standardmäßig einen vi-Puffer zur Bearbeitung der Ressourcendefinition.

oc delete

Mit dem Befehl `oc delete RESOURCE_TYPE name` wird eine Ressource aus einem OpenShift-Cluster entfernt. Beachten Sie, dass Sie über ein grundlegendes Verständnis der OpenShift-Architektur verfügen müssen, da das Löschen von verwalteten Ressourcen, wie beispielsweise Pods, neue Instanzen der automatisch erstellten Ressourcen zur Folge hat. Beim Löschen eines Projekts werden alle darin enthaltenen Ressourcen und Anwendungen ebenfalls gelöscht.

oc exec

Mit dem Befehl `oc exec CONTAINER_ID options` werden Befehle in einem Container ausgeführt. Verwenden Sie diesen Befehl, um interaktive und nicht interaktive Batch-Befehle im Rahmen eines Skripts auszuführen.

Beschriften von Ressourcen

Wenn Sie mit vielen Ressourcen in einem Projekt arbeiten, ist es oft nützlich, diese Ressourcen nach Anwendung, Umgebung oder anderen Kriterien zu gruppieren. Um diese Gruppen einzurichten, definieren Sie Bezeichnungen für die Ressourcen in Ihrem Projekt. Bezeichnungen sind ein Teil des Abschnitts `metadata` einer Ressource und werden als Schlüssel/Wert-Paare definiert, wie im folgenden Beispiel gezeigt:

```
apiVersion: v1
kind: Service
metadata:
...contents omitted...
labels: app: nexus template: nexus-persistent-template
name: nexus
...contents omitted...
```

Viele `oc`-Unterbefehle unterstützen eine `-l`-Option zum Verarbeiten von Ressourcen aus einer Bezeichnungsspezifikation. Für den Befehl `oc get` fungiert die Option `-l` als Selektor, um nur Objekte abzurufen, die eine entsprechende Bezeichnung haben:

```
$ 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         ...
```



Anmerkung

Obwohl jede Bezeichnung in Ressourcen eingesetzt werden kann, sind die `app`- und `template`-Schlüssel gängig für Bezeichnungen. Konventionsgemäß gibt der `app`-Schlüssel die Anwendung an, die sich auf diese Ressource bezieht. Der `template`-Schlüssel kennzeichnet alle Ressourcen, die von derselben Vorlage generiert werden, mit dem Namen der Vorlage.

Wenn Vorlagen zum Generieren von Ressourcen verwendet werden, sind Bezeichnungen besonders nützlich. Eine Vorlagenressource verfügt über einen `labels`-Abschnitt, der vom `metadata.labels`-Abschnitt getrennt ist. Im Abschnitt `labels` definierte Bezeichnungen gelten nicht für die Vorlage selbst, sondern werden jeder von der Vorlage generierten Ressource hinzugefügt:

```
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...
```

Im vorherigen Beispiel wird eine Vorlagenressource mit einer einzelnen Bezeichnung definiert: **maintainer: redhat**. Die Vorlage generiert eine Serviceressource mit drei Bezeichnungen: **app: nexus, template: nexus-persistent-template und version: 1**.



Literaturhinweise

Weitere Informationen über Pods und Services finden Sie im Abschnitt *Pods und Services* der Dokumentation zu OpenShift Container Platform:

Architektur

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/architecture/index

Weitere Informationen zur Erstellung von Images finden Sie in der Dokumentation zu OpenShift Container Platform:

Erstellen von Images

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/images/index

Details zu Bezeichnungen und Bezeichnungsselektoren sind im Abschnitt *Working with Kubernetes Objects* der Kubernetes-Dokumentation verfügbar:

Bezeichnungen und Selektoren

<https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/>

► Angeleitete Übung

Bereitstellen eines Datenbankservers auf OpenShift

In dieser Übung erstellen Sie mithilfe des Befehls `oc new-app` einen MySQL-Datenbankpod und stellen diesen auf OpenShift bereit.

Ergebnisse

Sie sollten in der Lage sein, einen MySQL-Datenbankpod auf OpenShift zu erstellen und bereitzustellen.

Bevor Sie Beginnen

Führen Sie auf `workstation` den folgenden Befehl zum Einrichten der Umgebung aus:

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

Anweisungen

- 1. Bereiten Sie die Umgebung für die praktische Übung vor.

- 1.1. Laden Sie die Konfiguration Ihrer Kursumgebung.

Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

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

- 1.2. Melden Sie sich beim OpenShift-Cluster an.

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

- 1.3. Erstellen Sie ein neues Projekt, das Ihren RHOCOP-Entwicklernamen enthält, für die Ressourcen, die Sie während dieser Übung erstellen werden:

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

- 2. Erstellen Sie anhand der Vorlage `mysql-persistent` mit dem Befehl `oc new-app` eine neue Anwendung.

Dieses Image erfordert, dass Sie die Option `-p` verwenden, um die Umgebungsvariablen `MYSQL_USER`, `MYSQL_PASSWORD`, `MYSQL_DATABASE`, `MYSQL_ROOT_PASSWORD` und `VOLUME_CAPACITY` festzulegen.

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

Verwenden Sie die Option `--template` mit dem Befehl `oc new-app`, um eine Vorlage mit persistentem Storage anzugeben. Dies verhindert, dass OpenShift das Image aus dem Internet abruft:

```
[student@workstation ~]$ oc new-app \
> --template=mysql-persistent \
> -p MYSQL_USER=user1 -p MYSQL_PASSWORD=mypa55 -p MYSQL_DATABASE=testdb \
> -p MYSQL_ROOT_PASSWORD=r0otpa55 -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. Verifizieren Sie, dass der MySQL-Pod erfolgreich erstellt wurde, und zeigen Sie die Details zum Pod und dessen Service an.
 - 3.1. Führen Sie den Befehl `oc status` aus, um den Status der neuen Anwendung anzuzeigen und zu verifizieren, dass die Bereitstellung des MySQL-Image erfolgreich war:

```
[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. Listen Sie die Pods aus diesem Projekt auf, um zu verifizieren, dass der MySQL-Pod bereit ist und ausgeführt wird:

NAME	READY	STATUS	RESTARTS	AGE
mysql-1-5vfn4	1/1	Running	0	109s

**Anmerkung**

Notieren Sie den Namen des ausgeführten Pods. Sie benötigen diese Informationen, um sich später beim MySQL-Datenbankserver anmelden zu können.

- 3.3. Führen Sie den Befehl `oc describe` aus, um weitere Details zum Pod anzuzeigen:

```
[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. Listen Sie die Services aus diesem Projekt auf und verifizieren Sie, dass für den Zugriff auf den MySQL-Pod ein Service erstellt wurde:

```
[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. Rufen Sie die Details des mysql-Services mit dem Befehl `oc describe` ab. Beachten Sie, dass der Standardtyp für Services `ClusterIP` ist:

```
[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:             172.30.151.91
Port:           3306-tcp  3306/TCP
TargetPort:     3306/TCP
Endpoints:     10.10.0.34:3306
Session Affinity: None
Events:         <none>
```

- 3.6. Führen Sie die Anforderungen für persistenten Storage in diesem Projekt auf:

```
[student@workstation ~]$ oc get pvc
NAME      STATUS    VOLUME                                     CAPACITY  ...
STORAGECLASS
mysql     Bound     pvc-e9bf0b1f-47df-4500-afb6-77e826f76c15  10Gi     ...  
standard
```

- 3.7. Rufen Sie die Details der mysql-PVC mit dem Befehl `oc describe` ab:

```
[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
Mounted By: mysql-1-5vfn4
```

- 3.8. Stellen Sie den Service bereit, indem Sie eine Route mit einem Standardnamen und dem vollständig qualifizierten Domain Name (FQDN) erstellen:

```
[student@workstation ~]$ oc expose service mysql
route.route.openshift.io/mysql exposed
[student@workstation ~]$ oc get routes
NAME      HOST/PORT          ... PORT
mysql     mysql-${RHT_OCP4_DEV_USER}-mysql... ... 3306-tcp
```

- 4. Stellen Sie eine Verbindung zum MySQL-Datenbankserver her, und verifizieren Sie, dass die Datenbank erstellt wurde.
- 4.1. Konfigurieren Sie auf dem Rechner `workstation` die Portweiterleitung zwischen `workstation` und dem Datenbank-Pod, der unter OpenShift mit Port 3306 ausgeführt wird. Das Terminal bleibt hängen, nachdem der Befehl ausgeführt wurde.

```
[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. Öffnen Sie auf dem Rechner `workstation` ein weiteres Terminal, und stellen Sie mithilfe des MySQL-Clients eine Verbindung zum MySQL-Server her.

```
[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: 8.0.21 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
```

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

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

- 4.3. Verifizieren Sie die Erstellung der testdb Datenbank.

```
mysql> show databases;  
-----  
| Database      |  
-----  
| information_schema |  
| testdb          |  
-----  
2 rows in set (0.00 sec)
```

- 4.4. Beenden Sie die MySQL-Eingabeaufforderung:

```
mysql> exit  
Bye
```

Schließen Sie das Terminal, und kehren Sie zum vorherigen zurück. Beenden Sie die Portweiterleitung. Drücken Sie dazu Strg+C.

```
Forwarding from 127.0.0.1:3306 -> 3306  
Forwarding from [::1]:3306 -> 3306  
Handling connection for 3306  
^C
```

- 5. Löschen Sie das Projekt, um alle Ressourcen im Projekt zu entfernen:

```
[student@workstation ~]$ oc delete project ${RHT_OCP4_DEV_USER}-mysql-openshift
```

Beenden

Führen Sie auf workstation das Skript lab openshift-resources finish aus, um diese praktische Übung abzuschließen.

```
[student@workstation ~]$ lab openshift-resources finish
```

Hiermit ist diese Übung beendet.

Erstellen von Routen

Ziele

In diesem Abschnitt wird beschrieben, wie Services mithilfe von OpenShift-Routen bereitgestellt werden.

Arbeiten mit Routen

Services erlauben den Netzwerzugriff zwischen Pods innerhalb einer OpenShift-Instanz. Routen ermöglichen Benutzern und Anwendungen den Netzwerzugriff auf Pods außerhalb der OpenShift-Instanz.

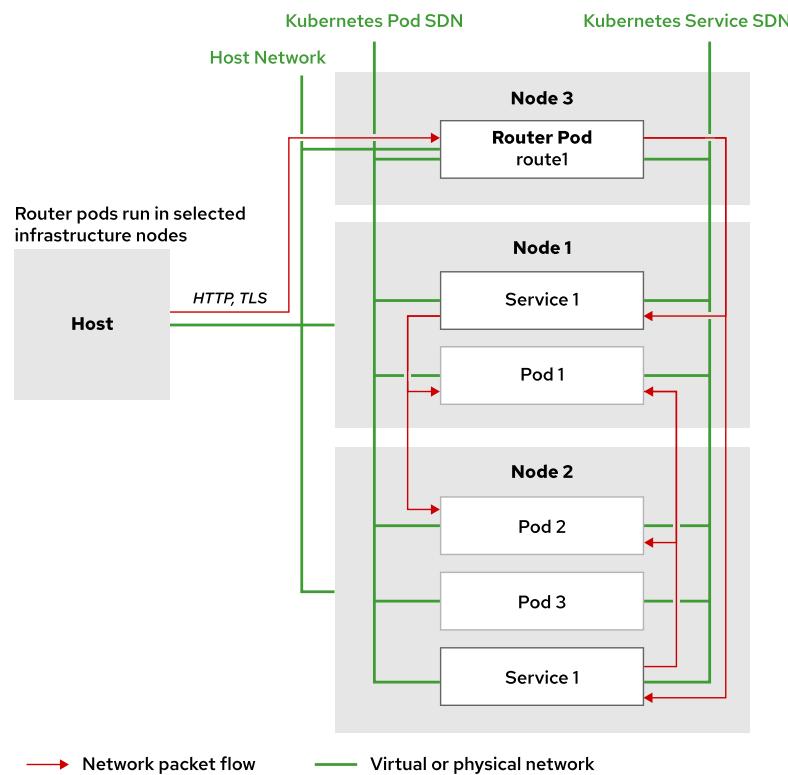


Abbildung 6.7: OpenShift-Routen und Kubernetes-Services

Eine Route stellt eine Verbindung zwischen einer öffentlichen IP-Adresse und einem DNS-Hostnamen zu einer internen Service-IP-Adresse her. Sie verwendet die Serviceressource, um die Endpunkte zu suchen, d. h. die vom Service bereitgestellten Ports.

OpenShift-Routen werden durch einen clusterweiten Routerservice implementiert, der eine containerisierte Anwendung im OpenShift-Cluster ausführt. OpenShift skaliert und repliziert Router-Pods wie jede andere OpenShift-Anwendung.



Anmerkung

Um die Leistung zu verbessern und die Latenz zu reduzieren, stellt der OpenShift-Router in der Praxis eine direkte Verbindung zu den Pods über das interne softwaredefinierte Netzwerk (SDN) her.

Der RouterService verwendet *HAProxy* als Standardimplementierung.

Ein wichtiger Aspekt für OpenShift-Administratoren ist die Tatsache, dass die für Routen konfigurierten öffentlichen DNS-Hostnamen auf die öffentlichen IP-Adressen der Knoten verweisen müssen, auf denen der Router ausgeführt wird. Router-Pods sind im Gegensatz zu regulären Anwendungs-Pods an die öffentlichen IP-Adressen ihrer Knoten gebunden, anstatt an das interne Pod-SDN.

Das folgende Beispiel zeigt eine Minimaldefinition einer Route in der JSON-Syntax:

```
{  
    "apiVersion": "v1",  
    "kind": "Route",  
    "metadata": {  
        "name": "quoteapp"  
    },  
    "spec": {  
        "host": "quoteapp.apps.example.com",  
        "to": {  
            "kind": "Service",  
            "name": "quoteapp"  
        }  
    }  
}
```

Die Attribute `apiVersion`, `kind` und `metadata` folgen den Kubernetes-Standardregeln zur Definition von Ressourcen. Der Wert `Route` für `kind` gibt an, dass dies eine Routenressource ist. Das Attribut `metadata.name` weist dieser spezifischen Route die Kennung `quoteapp` zu.

Ähnlich wie bei Pods und Services ist der Hauptteil das `spec`-Attribut. Dies ist ein Objekt, das die folgenden Attribute enthält:

- `host` ist eine Zeichenfolge, die den vollständig qualifizierten Domänennamen enthält, der der Route zugewiesen ist. DNS muss diesen FQDN in die IP-Adresse des OpenShift-Routers auflösen. Die Details zum Ändern der DNS-Konfiguration liegen nicht im Rahmen dieses Kurses.
- `to` ist ein Objekt, das die Ressource angibt, auf die diese Route verweist. In diesem Fall zeigt die Route auf einen OpenShift-Service, wobei der `name` auf `quoteapp` festgelegt ist.



Anmerkung

Die Namen der verschiedenen Ressourcenarten überschneiden sich nicht. Sie können durchaus eine Route mit dem Namen `quoteapp` verwenden, die auf einen Service mit demselben Namen, `quoteapp`, verweist.



Wichtig

Im Gegensatz zu Services, die zur Verknüpfung von Pod-Ressourcen mit spezifischen Bezeichnungen Selektoren verwenden, stellen Routen direkt eine Verknüpfung zum Service-Ressourcennamen her.

Erstellen von Routen

Führen Sie wie bei jeder anderen OpenShift-Ressource den Befehl `oc create` zum Erstellen von Ressourcen aus. Sie müssen eine JSON- oder YAML-Ressourcendefinitionsdatei, in der die Route definiert ist, für den Befehl `oc create` bereitstellen.

Mit dem Befehl `oc new-app` werden Routenressourcen nicht erzeugt, wenn Pods mittels Container-Images, Dockerfiles oder Anwendungsquellcode erstellt werden. Beim Ausführen des Befehls `oc new-app` ist nicht bekannt, ob der Zugriff auf den Pod außerhalb der OpenShift-Instanz zugelassen werden soll.

Eine andere Methode zur Erstellung einer Route ist die Verwendung des Befehls `oc expose service`, bei dem ein Service-Ressourcename als Eintrag weitergeleitet wird. Mithilfe der Option `--name` kann der Name der Routenressource gesteuert werden. Beispiel:

```
$ oc expose service quotedb --name quote
```

Standardmäßig generieren von `oc expose` erstellte Routen DNS-Namen im folgenden Format:
`route-name-project-name.default-domain`

Wobei Folgendes gilt:

- `route-name` ist der Name, welcher der Route zugewiesen ist. Wenn kein expliziter Name festgelegt ist, weist OpenShift der Route denselben Namen wie die Ursprungsressource zu (beispielsweise den Servicenamen).
- `project-name` ist der Name des Projekts, das die Ressource enthält.
- `default-domain` wird auf der OpenShift-Control Plane konfiguriert und entspricht der DNS-Platzhalter-Domain, die als Voraussetzung für die Installation von OpenShift aufgeführt ist.

Beispielsweise wird beim Erstellen einer Route mit dem Namen `quote` im Projekt mit dem Namen `test` über eine OpenShift-Instanz, wobei die Platzhalter-Domain `cloudapps.example.com` lautet, der FQDN `quote-test.cloudapps.example.com` ausgegeben.



Anmerkung

Der DNS-Server, der die Platzhalter-Domain hostet, kennt die Routing-Hostnamen nicht. Er löst lediglich alle Namen in die konfigurierten IP-Adressen auf. Nur der OpenShift-Router kennt die Routing-Hostnamen und behandelt jeden Namen wie einen virtuellen HTTP-Host. Der OpenShift-Router blockiert ungültige Hostnamen der Platzhalter-Domain, die keiner Route entsprechen, und gibt einen HTTP 404-Fehler aus.

Nutzen des standardmäßigen Routing-Services

Der standardmäßige Routing-Service wird als ein HAProxy-Pod implementiert. Router-Pods, Container und ihre Konfiguration können wie jede andere Ressource in einem OpenShift-Cluster überprüft werden:

```
$ oc get pod --all-namespaces -l app=router
NAMESPACE      NAME          READY STATUS RESTARTS AGE
openshift-ingress  router-default-746b5cfb65-f6sdm 1/1   Running 1        4d
```

Beachten Sie, dass Sie Informationen zum Standardrouter mit der zugehörigen Bezeichnung abfragen können, wie hier gezeigt wird.

Der Router wird standardmäßig im Projekt `openshift-ingress` bereitgestellt. Führen Sie den Befehl `oc describe pod` aus, um die Routing-Konfigurationsdetails abzurufen:

```
$ oc describe pod router-default-746b5cfb65-f6sdm
Name:           router-default-746b5cfb65-f6sdm
Namespace:      openshift-ingress
...
Containers:
  router:
    ...
    Environment:
      STATS_PORT:          1936
      ROUTER_SERVICE_NAMESPACE:  openshift-ingress
      DEFAULT_CERTIFICATE_DIR: /etc/pki/tls/private
      ROUTER_SERVICE_NAME:    default
      ROUTER_CANONICAL_HOSTNAME: apps.cluster.lab.example.com
...

```

Die Subdomain oder standardmäßige Domain, die in allen Standardrouten verwendet werden soll, bezieht ihren Wert aus dem Eintrag `ROUTER_CANONICAL_HOSTNAME`.



Literaturhinweise

Zusätzliche Informationen zur Architektur der Routen in OpenShift finden Sie in den Abschnitten *Architecture* und *Developer Guide* von

Dokumentation zu OpenShift Container Platform.

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/

► Angeleitete Übung

Bereitstellen eines Service als Route

In dieser Übung erstellen und implementieren Sie eine Anwendung in einem OpenShift-Cluster und stellen deren Dienst als Route zur Verfügung.

Ergebnisse

Sie sollten in der Lage sein, für eine bereitgestellte OpenShift-Anwendung einen Service als Route bereitzustellen.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

- 1. Bereiten Sie die Umgebung für die praktische Übung vor.

- 1.1. Laden Sie die Konfiguration Ihrer Kursumgebung.

Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

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

- 1.2. Melden Sie sich beim OpenShift-Cluster an.

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

- 1.3. Erstellen Sie ein neues Projekt, das Ihren RHOCOP-Entwicklernamen enthält, für die Ressourcen, die Sie während dieser Übung erstellen werden.

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-route
```

- 2. Erstellen Sie mit dem Source-to-Image aus dem Verzeichnis `php-helloworld` im Git-Repository unter [http://github.com/\\${RHT_OCP4_GITHUB_USER}/DO180-apps/](http://github.com/${RHT_OCP4_GITHUB_USER}/DO180-apps/) eine neue PHP-Anwendung.

- 2.1. Erstellen Sie mithilfe des Befehls `oc new-app` die PHP-Anwendung.



Wichtig

Im folgenden Beispiel wird ein umgekehrter Schrägstrich (\) verwendet, um anzugeben, dass die zweite Zeile eine Fortsetzung der ersten Zeile ist. Wenn Sie den umgekehrten Schrägstrich ignorieren möchten, können Sie den gesamten Befehl in einer Zeile eingeben.

```
[student@workstation ~]$ oc new-app \
> --image-stream php:7.3~https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps \
> --context-dir php-helloworld --name php-helloworld
--> Found image 688c0bd (2 months old) in image stream "openshift/php" under tag
"7.3" for "php:7.3"
...output omitted...
--> Creating resources ...
...output omitted...
--> Success
Build scheduled, use 'oc logs -f bc/php-helloworld' to track its progress.
Application is not exposed. You can expose services to the outside world by
executing one or more of the commands below:
  'oc expose svc/php-helloworld'
Run 'oc status' to view your app.
```

- 2.2. Warten Sie, bis die Anwendung den Erstellungs- und Bereitstellungsvorgang beendet hat, indem Sie den Fortschritt mithilfe des Befehls `oc get pods -w` überwachen:

```
[student@workstation ~]$ oc get pods -w
NAME           READY   STATUS    RESTARTS   AGE
php-helloworld-1-build   1/1     Running   0          7s
php-helloworld-598b4c66bc-g819w   0/1     Pending   0          0s
php-helloworld-598b4c66bc-g819w   0/1     Pending   0          0s
php-helloworld-598b4c66bc-g819w   0/1     ContainerCreating   0          0s
php-helloworld-1-build           0/1    Completed   0          33s
php-helloworld-598b4c66bc-g819w   0/1     ContainerCreating   0          2s
php-helloworld-598b4c66bc-g819w   0/1     ContainerCreating   0          3s
php-helloworld-598b4c66bc-g819w   1/1    Running   0          3s
^C
```

Ihre genaue Ausgabe kann sich in Namen, Status, Zeitpunkt und Reihenfolge unterscheiden. Der Container mit dem Status `Running` und einem zufälligen Suffix (im Beispiel `598b4c66bc-g819w`) enthält die Anwendung und zeigt an, dass sie betriebsbereit ist. Überwachen Sie alternativ die Build- und Bereitstellungsprotokolle mit den Befehlen `oc logs -f bc/php-helloworld` bzw. `oc logs -f php-helloworld-598b4c66bc-g819w`. Drücken Sie Strg + C, um den Befehl bei Bedarf zu beenden.

```
[student@workstation ~]$ oc logs -f bc/php-helloworld
Cloning "https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps" ...
Commit: f7cd8963ef353d9173c3a21dcccf402f3616840b (Initial commit, including all
apps previously in course)
...output omitted...
STEP 7: USER 1001
```

```
STEP 8: RUN /usr/libexec/s2i/assemble
--> Installing application source...
...output omitted...
Push successful
[student@workstation ~]$ oc logs -f php-helloworld-598b4c66bc-g819w
-> Cgroups memory limit is set, using HTTPD_MAX_REQUEST_WORKERS=136
=> sourcing 20-copy-config.sh ...
...output omitted...
[core:notice] [pid 1] AH000094: Command line: 'httpd -D FOREGROUND'
`^C`
```

Ihre genaue Ausgabe kann abweichen.

- 2.3. Überprüfen Sie den Service für diese Anwendung mithilfe des Befehls `oc describe`:

```
[student@workstation ~]$ oc describe svc/php-helloworld
Name:           php-helloworld
Namespace:      ${RHT_OCP4_DEV_USER}-route
Labels:          app=php-helloworld
                 app.kubernetes.io/component=php-helloworld
                 app.kubernetes.io/instance=php-helloworld
Annotations:    openshift.io/generated-by: OpenShiftNewApp
Selector:        deployment=php-helloworld
Type:           ClusterIP
IP:             172.30.228.124
Port:           8080-tcp  8080/TCP
TargetPort:     8080/TCP
Endpoints:      10.10.0.35:8080
Port:           8443-tcp  8443/TCP
TargetPort:     8443/TCP
Endpoints:      10.10.0.35:8443
Session Affinity: None
Events:         <none>
```

Die in der Ausgabe des Befehls angezeigte IP-Adresse kann variieren.

- 3. Stellen Sie den Service bereit, durch den eine Route erstellt wird. Verwenden Sie den Standardnamen und den vollständig qualifizierten Domain Name (FQDN) für die Route:

```
[student@workstation ~]$ oc expose svc/php-helloworld
route.route.openshift.io/php-helloworld exposed
[student@workstation ~]$ oc describe route
Name:           php-helloworld
Namespace:      ${RHT_OCP4_DEV_USER}-route
Created:        5 seconds ago
Labels:          app=php-helloworld
                 app.kubernetes.io/component=php-helloworld
                 app.kubernetes.io/instance=php-helloworld
Annotations:    openshift.io/host.generated=true
Requested Host: php-helloworld-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDC...
               exposed on router default (host ${RHT_OCP4_WILDCARD_DOMAIN}) 5 seconds ago
Path:           <none>
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port:  8080-tcp
```

```
Service:      php-helloworld
Weight:      100 (100%)
Endpoints:   10.10.0.35:8080, 10.10.0.35:8443
```

- 4. Greifen Sie von einem Host außerhalb des Clusters auf den Service zu, um zu verifizieren, dass der Service und die Route funktionieren.

```
[student@workstation ~]$ curl \
> php-helloworld-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.20
```



Anmerkung

Die Ausgabe der PHP-Anwendung hängt vom tatsächlichen Code im Git-Repository ab. Sie kann anders sein, wenn Sie den Code in den vorherigen Abschnitten aktualisiert haben.

Beachten Sie Folgendes: Der FQDN setzt sich standardmäßig aus dem Anwendungsnamen und dem Projektnamen zusammen. Der Rest des FQDN, die Subdomain, wird bei der Installation von OpenShift definiert.

- 5. Ersetzen Sie diese Route durch eine Route namens xyz.

- 5.1. Löschen Sie die aktuelle Route:

```
[student@workstation ~]$ oc delete route/php-helloworld
route.route.openshift.io "php-helloworld" deleted
```



Anmerkung

Das Löschen der Route ist optional. Sie können mehrere Routen für denselben Service besitzen, vorausgesetzt, sie haben unterschiedliche Namen.

- 5.2. Erstellen Sie für den Service eine Route mit dem Namen \${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:          ${RHT_OCP4_DEV_USER}-xyz
Namespace:     ${RHT_OCP4_DEV_USER}-route
Created:       5 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: ${RHT_OCP4_DEV_USER}-xyz-${RHT_OCP4_DEV_USER}-route.${RHT_...
               exposed on router default (host ${RHT_OCP4_WILDCARD_DOMAIN}) 4 seconds ago
Path:          <none>
```

```
TLS Termination: <none>
Insecure Policy: <none>
Endpoint Port: 8080-tcp

Service: php-helloworld
Weight: 100 (100%)
Endpoints: 10.10.0.35:8080, 10.10.0.35:8443
```

Beachten Sie den neuen FQDN, der basierend auf dem neuen Routennamen generiert wurde. Sowohl der Routenname als auch der Projektname enthalten Ihren Benutzernamen. Daher wird er zweimal im Routen-FQDN angezeigt.

- 5.3. Erstellen Sie eine HTTP-Anfrage unter Verwendung des FQDN über Port 80:

```
[student@workstation ~]$ curl \
> ${RHT_OCP4_DEV_USER}-xyz-${RHT_OCP4_DEV_USER}-route.${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.20
```

Beenden

Führen Sie auf workstation das Skript `lab openshift-routes finish` aus, um diese Übung zu beenden.

```
[student@workstation ~]$ lab openshift-routes finish
```

Hiermit ist die angeleitete Übung beendet.

Erstellen von Anwendungen mit Source-to-Image

Ziele

Nach Abschluss dieses Kapitels sollten die Teilnehmer eine Anwendung mithilfe der Funktion Source-to-Image (S2I) der OpenShift Container Platform bereitstellen können.

Der S2I-Prozess (Source-to-Image)

Source-to-Image (S2I) ist ein Tool, das die Erstellung von Container-Images über Anwendungsquellcode erleichtert. Dieses Tool verwendet den Quellcode einer Anwendung von einem Git-Repository, fügt den Quellcode unter Berücksichtigung der gewünschten Sprache und des gewünschten Frameworks in einen Basis-Container ein und erstellt ein neues Container-Image, das die zusammengestellte Anwendung ausführt.

In dieser Abbildung werden die Ressourcen angezeigt, die vom Befehl `oc new-app <source>` erstellt werden, wenn das Argument ein Repository für den Anwendungsquellcode darstellt. Beachten Sie, dass S2I außerdem eine Deployment-Ressource und alle abhängigen Ressourcen erstellt:

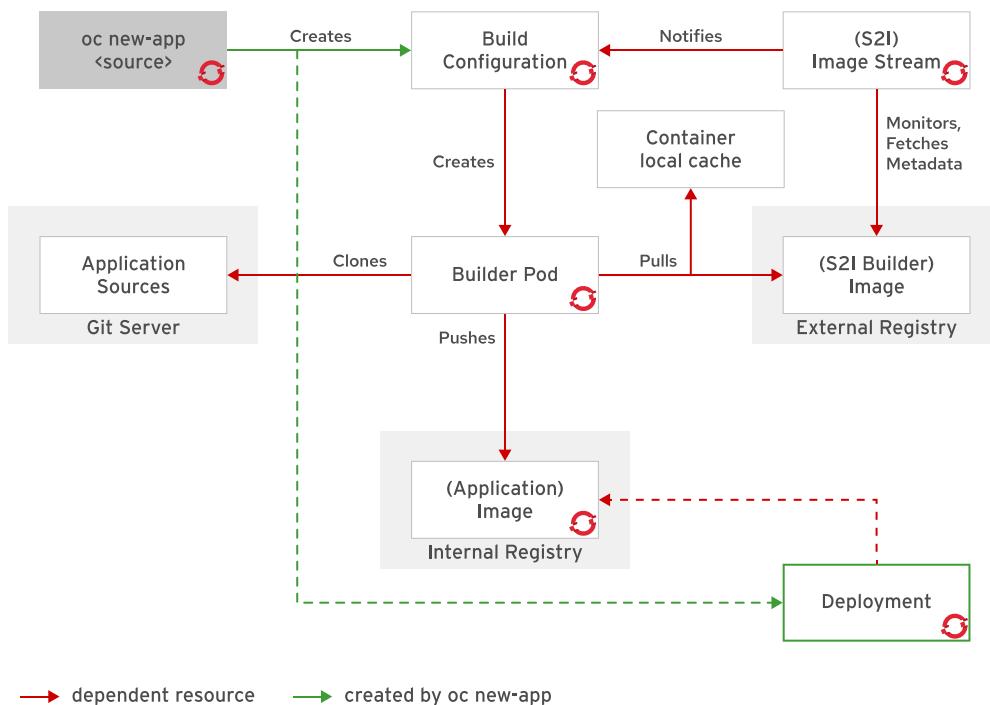


Abbildung 6.8: Deployment-Ressource und zugehörige Ressourcen

S2I ist die primäre Strategie zur Erstellung von Anwendungen in OpenShift Container Platform. Folgende Gründe sprechen für die Verwendung von Quell-Builds:

- Benutzerproduktivität: Entwickler müssen Dockerfiles und Betriebssystembefehle wie `yum install` nicht erlernen. Sie arbeiten mit ihren Standardtools für Programmiersprachen.

- Patchen: Mit S2I können alle Anwendungen konsistent neu erstellt werden, wenn ein Basis-Image infolge eines Sicherheitsproblems einen Patch benötigt. Beispiel: Wenn in einem PHP-Basis-Image ein Sicherheitsproblem festgestellt wird, dann werden beim Aktualisieren dieses Image mithilfe von Sicherheitspatches alle Anwendungen aktualisiert, die dieses Image als Basis-Image verwenden.
- Geschwindigkeit: Mit S2I kann während des Fertigungsprozesses eine große Zahl komplexer Vorgänge durchgeführt werden, ohne in jedem Schritt eine neue Ebene erstellen zu müssen. Dies ermöglicht schnellere Builds.
- Ökosystem: S2I ermöglicht ein gemeinsames Image-Ökosystem, in dem Basis-Images und Skripts angepasst und in zahlreichen Anwendungen erneut verwendet werden können.

Beschreiben von Image-Streams

OpenShift stellt neue Versionen von Benutzeranwendungen in kürzester Zeit in Pods bereit. Um eine neue Anwendung zu erstellen, ist neben dem Anwendungsquellcode ein Basis-Image (das S2I-Builder-Image) erforderlich. Bei der Aktualisierung einer dieser beiden Komponenten erstellt OpenShift ein neues Container-Image. Pods, die mit dem älteren Container-Image erstellt wurden, werden durch Pods mit dem neuen Image ersetzt.

Dass ein Container-Image aktualisiert werden muss, wenn sich Anwendungscode ändert, ist offensichtlich. Es ist jedoch möglicherweise nicht offensichtlich, dass bereitgestellte Pods ebenfalls aktualisiert werden müssen, wenn sich das Builder-Image ändert.

Die Image-Stream-Ressource ist eine Konfiguration, die bestimmten Container-Images, die Image Stream-Tags zugeordnet sind, einen Alias für diese Container-Images nennt. OpenShift erstellt Anwendungen anhand eines Image-Streams. Das OpenShift-Installationsprogramm füllt standardmäßig während der Installation mehrere Image-Streams aus. Führen Sie den Befehl `oc get` aus, um die verfügbaren Image-Streams wie folgt zu bestimmen:

```
$ oc get is -n openshift
NAME          IMAGE REPOSITORY           TAGS
cli           ...svc:5000/openshift/cli    latest
dotnet        ...svc:5000/openshift/dotnet  2.1,...,3.1-e17,latest
dotnet-runtime ...svc:5000/openshift/dotnet-runtime 2.1,...,3.1-e17,latest
httpd         ...svc:5000/openshift/httpd   2.4,2.4-e17,2.4-e18,latest
jenkins       ...svc:5000/openshift/jenkins  2,latest
mariadb       ...svc:5000/openshift/mariadb  10.3,10.3-e17,10.3-e18,latest
mongodb       ...svc:5000/openshift/mongodb  3.6,latest
mysql         ...svc:5000/openshift/mysql   8.0,8.0-e17,8.0-e18,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-e17
php           ...svc:5000/openshift/php    7.2-ubi8,...,7.3-ubi8,latest
postgresql   ...svc:5000/openshift/postgresql 10,10-e17,...,12-e17,12-e18
python        ...svc:5000/openshift/python  2.7,2.7-ubi7,...,3.6-ubi8,3.8
redis         ...svc:5000/openshift/redis   5,5-e17,5-e18,latest
ruby          ...svc:5000/openshift/ruby   2.5,2.5-ubi7,...,2.6,2.6-ubi7
...
```



Anmerkung

In OpenShift-Instanzen können mehr oder weniger Image-Streams vorhanden sein. Dies hängt von lokalen Ergänzungen und OpenShift-Versionen ab.

OpenShift erkennt, wann ein Image-Stream geändert wird, und führt der Änderung entsprechende Maßnahmen durch. Wenn im Image `rhel8/nodejs-10` ein Sicherheitsproblem auftritt, kann es im Image-Repository aktualisiert werden. OpenShift kann dann die Neuerstellung des Anwendungscodes automatisch auslösen.

Eine Organisation wird wahrscheinlich mehrere unterstützte S2I-Basis-Images von Red Hat auswählen, kann aber auch eigene Basis-Images erstellen.

Erstellen einer Anwendung mit S2I und der CLI

Die Erstellung einer Anwendung mit S2I kann über die CLI von OpenShift durchgeführt werden.

Anwendungen können unter Verwendung des S2I-Prozesses und des Befehls `oc new-app` über die CLI erstellt werden.

```
$ oc new-app php~http://my.git.server.com/my-app ①②  
--name=myapp ③
```

- ① Der in diesem Prozess verwendete Image-Stream wird links neben der Tilde (~) angezeigt.
- ② Die URL nach der Tilde gibt den Speicherort des Git-Repositories des Quellcodes an.
- ③ Legen Sie den Anwendungsnamen fest.



Anmerkung

Anstatt die Tilde zu verwenden, können Sie den Image-Stream über die Option `-i` oder `--image-stream` für die Vollversion festlegen.

```
$ oc new-app -i php http://services.lab.example.com/app --name=myapp
```

Der Befehl `oc new-app` ermöglicht die Erstellung von Anwendungen mithilfe des Quellcodes über ein lokales oder Remote-Git-Repository. Wenn nur ein Quell-Repository angegeben wird, versucht der Befehl `oc new-app` den korrekten Image-Stream für die Erstellung der Anwendung zu identifizieren. Zusätzlich zum Anwendungscode kann S2I Dockerfiles für die Erstellung eines neuen Images identifizieren und verarbeiten.

Im folgenden Beispiel wird eine Anwendung unter Verwendung des Git-Repositorys im aktuellen Verzeichnis erstellt:

```
$ oc new-app .
```



Wichtig

Bei Verwendung eines lokalen Git-Repositorys muss das Repository aus einem Remote-Speicherort stammen, der auf eine URL verweist, auf die über die OpenShift-Instanz zugegriffen werden kann.

Es ist auch möglich, eine Anwendung mit einem Git-Remote-Repository und Kontext-Unterverzeichnis zu erstellen:

```
$ oc new-app https://github.com/openshift/sti-ruby.git \
--context-dir=2.0/test/puma-test-app
```

Darüber hinaus kann eine Anwendung mit einem Git-Remote-Repository und einem spezifischen Verweis auf den Branch erstellt werden.

```
$ oc new-app https://github.com/openshift/ruby-hello-world.git#beta4
```

Wenn im Befehl kein Image-Stream angegeben ist, versucht new-app anhand des Vorhandenseins bestimmter Dateien im Stammverzeichnis des Repositorys zu ermitteln, welcher Sprachen-Builder verwendet werden soll:

Sprache	Ressourcen
Ruby	RakefileGemfile, config.ru
Java EE	pom.xml
Node.js	app.json package.json
PHP	index.php composer.json
Python	requirements.txt config.py
Perl	index.pl cpanfile

Nach Ermittlung der Sprache sucht der Befehl new-app nach Image-Stream-Tags, welche die ermittelte Sprache unterstützen, bzw. nach einem Image-Stream, der mit dem Namen der ermittelten Sprache übereinstimmt.

Erstellen Sie eine JSON-Ressourcendefinitionsdatei. Verwenden Sie dazu den Parameter -o json und die Ausgabeumleitung:

```
$ oc -o json new-app php~http://services.lab.example.com/app \
> --name=myapp > s2i.json
```

Diese JSON-Definitionsdatei erstellt eine Liste mit Ressourcen. Die erste Ressource ist der Image-Stream:

```
...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...
```

- ① Definieren Sie den Ressourcentyp des Image-Streams.
- ② Nennen Sie den Image-Stream `myapp`.

Die Build-Konfiguration (BC) ist für die Definition von Eingabeparametern und Auslösern zuständig, die ausgeführt werden, um den Quellcode in ein ausführbares Image umzuwandeln. Die Build-Konfiguration (BC) ist die zweite Ressource. Im folgenden Beispiel erhalten Sie eine Übersicht über die Parameter, die von OpenShift zur Erstellung eines ausführbaren Images verwendet werden.

```
...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** Definieren Sie den Ressourcentyp `BuildConfig`.
- ②** Legen Sie `myapp` als Namen für die `BuildConfig` fest.
- ③** Definieren Sie die Adresse zum Git-Repository des Quellcodes.
- ④** Definieren Sie die Strategie zur Verwendung von S2I.
- ⑤** Definieren Sie das Builder-Image als Image-Stream `php:7.3`.
- ⑥** Geben Sie dem Ausgabe-Image-Stream den Namen `myapp:latest`.

Die dritte Ressource ist das Bereitstellungsobjekt, die für die Anpassung des Bereitstellungsprozesses in OpenShift zuständig ist. Es kann Parameter und Auslöser enthalten, die für die Erstellung neuer Container-Instanzen erforderlich sind und in einem Replikationscontroller von Kubernetes übersetzt werden. Einige der Features, die von `Deployment`-Objekten zur Verfügung gestellt werden, sind:

- Vom Benutzer anpassbare Strategien für den Übergang von bestehenden Bereitstellungen zu neuen Bereitstellungen

- Besitz so vieler aktiver Replikate, wie gewünscht und möglich ist
- Die Replikations skalierung hängt von der Größe alter und neuer Replikatsätze ab.

```
...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:\\
                atest\\\"},\\\"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 Definieren Sie den Ressourcentyp Deployment.
- ➋ Legen Sie myapp als Namen für den Ressourcentyp Deployment fest.
- ➌ Ein Trigger für Konfigurationsänderungen bewirkt, dass bei jeder Änderung der Replikationscontroller-Vorlage eine neue Bereitstellung erstellt wird.
- ➍ Ein Trigger für Image-Änderungen bewirkt, dass jedes Mal, wenn eine neue Version des Images myapp:latest im Repository verfügbar gemacht wird, eine neue Bereitstellung erstellt wird.
- ➎ Definiert das bereitzustellende Container-Image: myapp:latest.
- ➏ Gibt die Container-Ports an.

Das letzte Element ist der Service (dies wurde bereits in vorherigen Kapiteln behandelt):

```
...output omitted...
{
    "kind": "Service",
    "apiVersion": "v1",
    "metadata": {
        "name": "myapp",
        "creationTimestamp": null,
        "labels": {
            "app": "myapp"
            "app.kubernetes.io/component": "myapp",
            "app.kubernetes.io/instance": "myapp"
        },
        "annotations": {
            "openshift.io/generated-by": "OpenShiftNewApp"
        }
    },
    "spec": {
        "ports": [
            {
                "name": "8080-tcp",
                "protocol": "TCP",
                "port": 8080,
                "targetPort": 8080
            },
            {
                "name": "8443-tcp",
                ...
            }
        ]
    }
}
```

```

        "protocol": "TCP",
        "port": 8443,
        "targetPort": 8443
    }
],
"selector": {
    "deployment": "myapp"
}
},
"status": {
    "loadBalancer": {}
}
}

```



Anmerkung

Der Befehl `oc new-app` erstellt standardmäßig keine Route. Nach Erstellung der Anwendung kann eine Route erstellt werden. Bei Verwendung der Webkonsole wird jedoch automatisch eine Route erstellt, da diese eine Vorlage verwendet.

Nach Erstellung einer neuen Anwendung wird der Build-Prozess gestartet. Führen Sie den Befehl `oc get builds` aus, um eine Liste der Anwendungs-Builds anzuzeigen:

```
$ oc get builds
NAME          TYPE      FROM      STATUS     STARTED      DURATION
php-helloworld-1  Source   Git@9e17db8  Running  13 seconds ago
```

OpenShift ermöglicht die Anzeige von Build-Protokollen. Der folgende Befehl zeigt die letzten Zeilen des Build-Protokolls an:

```
$ oc logs build/myapp-1
```



Wichtig

Wenn der Build noch nicht ausgeführt wird (Running) bzw. der s2i-build-Pod noch nicht bereitgestellt wurde, gibt der oben gezeigte Befehl eine Fehlermeldung aus. Warten Sie ein paar Sekunden und versuchen Sie es erneut.

Lösen Sie mit dem Befehl `oc start-build build_config_name` einen neuen Build aus:

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

```
$ oc start-build myapp
build "myapp-2" started
```

Beziehung zwischen Build und Deployment

Der Pod **BuildConfig** ist für die Erstellung der Images in OpenShift zuständig und sendet diese an die interne Container-Registry. Quellcode- oder Inhaltsaktualisierungen erfordern in der Regel einen neuen Build, um sicherzustellen, dass das Image aktualisiert wird.

Der Pod **Deployment** ist für die Bereitstellung von Pods in OpenShift zuständig. Das Ergebnis der Ausführung eines **Deployment**-Pods ist die Erstellung von Pods mit den Images, die in der internen Container-Registry bereitgestellt wurden. Alle ausgeführten Pods können entfernt werden, je nachdem, wie die **Deployment**-Ressource festgelegt ist.

Die **BuildConfig**- und **Deployment**-Ressourcen interagieren nicht direkt miteinander. Die **BuildConfig**-Ressource erstellt oder aktualisiert ein Container-Image. Die **Deployment**-Ressource reagiert auf dieses neue Image bzw. das aktualisierte Image und erstellt Pods über das Container-Image.



Literaturhinweise

Source-to-Image-Build (S2I)

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/builds/build-strategies#builds-strategy-s2i-build_understanding-image-builds

S2I-GitHub-Repository

<https://github.com/openshift/source-to-image>

► Angeleitete Übung

Erstellen einer containerisierten Anwendung mit Source-to-Image

In dieser Übung erstellen Sie eine Anwendung über den Quellcode und stellen die Anwendung in einem OpenShift-Cluster bereit.

Ergebnisse

Es werden folgende Fähigkeiten vermittelt:

- Erstellung einer Anwendung über den Quellcode mithilfe der Befehlszeilenschnittstelle von OpenShift
- Verifizierung der erfolgreichen Anwendungsbereitstellung mithilfe der Befehlszeilenschnittstelle von OpenShift

Bevor Sie Beginnen

Führen Sie den folgenden Befehl aus, um die relevanten Lab-Dateien herunterzuladen und die Umgebung zu konfigurieren:

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

Anweisungen

- 1. Überprüfen Sie den PHP-Quellcode für die Beispielanwendung, und erstellen und übertragen Sie einen neuen Branch namens `s2i`, der während dieser Übung verwendet werden soll.
- 1.1. Wechseln Sie zu Ihrem lokalen Klon des Git-Repositorys `D0180-apps`, und checken Sie den Branch `master` des Kurs-Repositorys aus, um sicherzustellen, dass Sie diese Übung von einem als fehlerfrei bekannten Zustand aus starten:

```
[student@workstation openshift-s2i]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 1.2. Erstellen Sie einen neuen Branch, um alle Änderungen zu speichern, die Sie während dieser Übung vornehmen:

```
[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. Überprüfen Sie den PHP-Quellcode der Anwendung im Ordner `php-helloworld`.

Öffnen Sie die Datei `index.php` im Ordner `/home/student/D0180-apps/php-helloworld`:

```
<?php  
print "Hello, World! php version is " . PHP_VERSION . "\n";  
?>
```

Die Anwendung implementiert eine einfache Antwort, die die ausgeführte PHP-Version zurückgibt.

► **2.** Bereiten Sie die Umgebung für die praktische Übung vor.

2.1. Laden Sie die Konfiguration Ihrer Kursumgebung.

Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

```
[student@workstation D0180-apps]$ source /usr/local/etc/ocp4.config
```

2.2. Melden Sie sich beim OpenShift-Cluster an.

```
[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. Erstellen Sie ein neues Projekt, das Ihren RHOCOP-Entwicklernamen enthält, für die Ressourcen, die Sie während dieser Übung erstellen werden:

```
[student@workstation D0180-apps]$ oc new-project ${RHT_OCP4_DEV_USER}-s2i
```

► **3.** Erstellen Sie eine neue PHP-Anwendung mit Source-to-Image aus dem Verzeichnis `php-helloworld`. Verwenden Sie dazu den Branch `s2i`, den Sie im vorherigen Schritt in Ihrem Fork des Git-Repositorys „D0180-apps“ erstellt haben.

3.1. Erstellen Sie mithilfe des Befehls `oc new-app` die PHP-Anwendung.



Wichtig

Im folgenden Beispiel wird das Nummernzeichen (#) verwendet, um einen spezifischen Branch aus dem Git-Repository auszuwählen, in diesem Fall den im vorherigen Schritt erstellten Branch `s2i`.

```
[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. Warten Sie, bis die Erstellung abgeschlossen und die Anwendung bereitgestellt wurde. Verifizieren Sie, dass der Erstellungsprozess mit dem Befehl `oc get pods` gestartet wird.

```
[student@workstation openshift-s2i]$ oc get pods
NAME             READY   STATUS    RESTARTS   AGE
php-helloworld-1-build   1/1     Running   0          5s
```

- 3.3. Überprüfen Sie die Protokolle für diesen Build. Verwenden Sie den Build-Pod-Namen für diesen Build `php-helloworld-1-build`.

```
[student@workstation D0180-apps]$ oc logs --all-containers \
> -f php-helloworld-1-build
...output omitted...

Writing manifest to image destination
Storing signatures
Generating dockerfile with builder image image-registry.openshift-image-...
php@sha256:3206...37b4
Adding transient rw bind mount for /run/secrets/rhsm
STEP 1: FROM image-registry.openshift-image-registry.svc:5000...
...output omitted...

STEP 8: RUN /usr/libexec/s2i/assemble
...output omitted...

Pushing image .../php-helloworld:latest ...
Getting image source signatures
...output omitted...

Writing manifest to image destination
Storing signatures
Successfully pushed .../php-
helloworld@sha256:3f1cdb278548c7f24429e2469c51ae35482d54e2616d596ab6fb59d6b432c454
Push successful
Cloning "https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps" ...
Commit: 9a042f7e3650ef38ad07af83b74f57c7a7d1820c (Added start up script)
...output omitted...
```

Beachten Sie, dass der Klon des Git-Repositorys der erste Schritt des Build-Prozesses ist. Als Nächstes erstellt der Source-to-Image-Prozess ein neues Image mit dem Namen `s2i/php-helloworld:latest`. Der letzte Schritt im Build-Prozess besteht darin, dieses Image an die private Registry von OpenShift zu senden.

- 3.4. Überprüfen Sie die Details der Bereitstellungskonfiguration (Deployment) für diese Anwendung:

```
[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
```

```

Annotations:           app.kubernetes.io/instance=php-helloworld
                      deployment.kubernetes.io/revision: 2
                      image.openshift.io/triggers:
                        [{"from":{"kind":"ImageStreamTag","name":"php-
helloworld:latest"},"fieldPath":"spec.template.spec.containers[?(@.name==\"php-
helloworld\")]..."}
Selector:            openshift.io/generated-by: OpenShiftNewApp
Replicas:             deployment=php-helloworld
                      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. Fügen Sie eine Route hinzu, um die Anwendung zu testen:

```
[student@workstation DO180-apps]$ oc expose service php-helloworld \
> --name ${RHT_OCP4_DEV_USER}-helloworld
route.route.openshift.io/${RHT_OCP4_DEV_USER}-helloworld exposed
```

3.6. Suchen Sie die mit der neuen Route verknüpfte URL:

```
[student@workstation DO180-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. Testen Sie die Anwendung. Senden Sie dazu eine HTTP GET-Anforderung an die URL, die Sie im vorherigen Schritt abgerufen haben:

```
[student@workstation DO180-apps]$ curl -s \
> ${RHT_OCP4_DEV_USER}-helloworld-${RHT_OCP4_DEV_USER}-s2i.\
> ${RHT_OCP4_WILDCARD_DOMAIN}
Hello, World! php version is 7.3.20
```

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

- 4. Überprüfen Sie die anfänglichen Anwendung-Builds, indem Sie die Anwendung im zugehörigen Git-Repository ändern und die entsprechenden Befehle ausführen, um ein neues Source-to-Image-Build zu starten.

- 4.1. Geben Sie das Quellcodeverzeichnis ein.

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

- 4.2. Bearbeiten Sie die Datei `index.php` wie folgt:

```
<?php  
print "Hello, World! php version is " . PHP_VERSION . "\n";  
print "A change is a coming!\n";  
?>
```

Speichern Sie die Datei.

- 4.3. Übernehmen Sie die Änderungen und senden Sie den Code zurück an das externe Git-Repository.

```
[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. Starten Sie einen neuen S2I-Buildprozess und warten Sie, bis die Erstellung und Bereitstellung abgeschlossen ist:

```
[student@workstation php-helloworld]$ oc start-build php-helloworld  
build.build.openshift.io/php-helloworld-2 started  
[student@workstation php-helloworld]$ oc logs php-helloworld-2-build -f  
...output omitted...  
  
Successfully pushed .../php-helloworld:latest@sha256:74e757a4c0edaeda497dab7...  
Push successful
```

**Anmerkung**

Nach dem Start des Builds kann es einige Sekunden dauern, bis die Protokolle verfügbar sind. Wenn der vorherige Befehl fehlschlägt, warten Sie ein wenig, und versuchen Sie es erneut.

- 4.5. Führen Sie nach Abschluss des zweiten Builds den Befehl `oc get pods` aus, um zu verifizieren, dass die neue Version der Anwendung ausgeführt wird.

```
[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. Testen Sie, ob die Anwendung den neuen Inhalt bereitstellt:

```
[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 7.3.20
A change is a coming!
```

Beenden

Führen Sie auf `workstation` das Skript `lab openshift-s2i finish` aus, um diese praktische Übung abzuschließen.

```
[student@workstation php-helloworld]$ lab openshift-s2i finish
```

Hiermit ist die angeleitete Übung beendet.

Erstellung von Anwendungen mit der OpenShift-Webkonsole

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Erstellung von Anwendungen mit der OpenShift Web Console
- Verwaltung und Überwachung des Build-Zyklus einer Anwendung
- Überprüfung der Ressourcen für eine Anwendung

Zugriff auf die OpenShift Web Console

Mit der OpenShift Web Console können Benutzer viele derselben Aufgaben wie der OpenShift-Befehlszeilen-Client ausführen. Sie können Projekte erstellen, Projekten Anwendungen hinzufügen, Anwendungsressourcen anzeigen und Anwendungskonfigurationen nach Bedarf bearbeiten. Die OpenShift Web Console wird als ein oder mehrere Pods ausgeführt, wobei jeder Pod in einer Control Plane ausgeführt wird.

Obwohl der Befehlszeilen-Client vielseitiger ist als die Webkonsole, können Sie feststellen, dass die visuelle Darstellung der Konzepte in OpenShift hilfreich ist.

Die Webkonsole wird in einem Webbrower ausgeführt.

Die Standard-URL hat das Format `https://console-openshift-console.{wildcard DNS domain for the RHOC cluster}/`. OpenShift generiert standardmäßig ein selbstsigniertes Zertifikat für die Webkonsole. Sie müssen für den Zugriff dieses Zertifikat als vertrauenswürdig einstufen.

Die Web Console verwendet eine REST-API zur Kommunikation mit dem OpenShift-Cluster. Standardmäßig wird mit einem anderen DNS-Namen und einem selbstsignierten Zertifikat auf den REST-API-Endpunkt zugegriffen. Sie müssen diesem Zertifikat auch für den REST-API-Endpunkt vertrauen.

Nachdem Sie die beiden OpenShift-Zertifikate als vertrauenswürdig eingestuft haben, erfordert die Konsole eine Authentifizierung, um fortfahren zu können.

Verwalten von Projekten

Nach der erfolgreichen Anmeldung wird auf der Seite **Home → Projects** eine Liste der Projekte angezeigt, auf die Sie zugreifen können. Auf dieser Seite können Sie ein Projekt erstellen, bearbeiten oder löschen.

Wenn Sie über die Berechtigung zum Anzeigen von Cluster-Metriken verfügen, werden Sie stattdessen zur **Home → Overview**-Seite umgeleitet. Diese Seite zeigt allgemeine Informationen und Metriken zum Cluster. Das Menüelement „> Overview“ ist für Benutzer ohne Berechtigung zur Anzeige von Cluster-Metriken verborgen.

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

Abbildung 6.9: OpenShift Web Console-Startseite

Das Ellipsensymbol am Ende jeder Zeile enthält ein Menü mit Projektaktionen. Wählen Sie den entsprechenden Eintrag aus, um das gewünschte Projekt zu bearbeiten oder zu löschen.

Wenn Sie in dieser Ansicht auf eine Projektverknüpfung klicken, werden Sie zur Seite **Project Status** weitergeleitet. Diese Seite zeigt alle Anwendungen, die in diesem Projektbereich erstellt wurden.

Navigieren in der Web Console

Auf der linken Seite der Web Console befindet sich ein Navigationsmenü. Das Menü enthält zwei Perspektiven: **Administrator** und **Developer**. Jedes Element im Menü kann erweitert werden, um Zugriff auf eine Reihe verwandter Verwaltungsfunktionen zu erhalten. Wenn die Perspektive **Administrator** ausgewählt ist, werden folgende Elemente angezeigt:

Home

Mit dem Menü „Home“ können Benutzer schnell auf Projekte und Ressourcen zugreifen. Mit diesem Menü können Sie Projekte durchsuchen und verwalten, Cluster-Ressourcen suchen oder durchsuchen und Cluster-Ereignisse überprüfen.

Operatoren

Der **OperatorHub** ist ein Katalog, mit dem Sie Operatoren im Cluster ermitteln, suchen und installieren können. Nach der Installation eines Operators können Sie mit der Option **Installed Operators** den Operator verwalten oder nach anderen installierten Operatoren suchen.



Anmerkung

Die neuesten Versionen von Kubernetes implementieren viele Controller als Operatoren. Operatoren sind Kubernetes-Plug-in-Komponenten, die auf Clusterereignisse reagieren und den Status von Ressourcen steuern können. Operatoren und das CoreOS Operator Framework werden in diesem Dokument nicht beschrieben.

Workloads

Diese Optionen ermöglichen die Verwaltung verschiedener Arten von Kubernetes- und OpenShift-Ressourcen wie Pods und Bereitstellungen. Andere erweiterte Bereitstellungsoptionen, auf die Sie über dieses Menü zugreifen können, wie Konfigurationskarten, Geheimnisse und Cron-Jobs, werden in diesem Kurs nicht behandelt.

Networking

Dieses Menü enthält Optionen zum Verwalten von OpenShift-Ressourcen, die sich auf den Anwendungszugriff, wie z. B. Services und Routen, für ein Projekt auswirken. Es gibt auch andere Optionen zum Konfigurieren einer OpenShift-Netzwerkrichtlinie oder eines Ingress-Elements. Diese werden in den Themen dieses Kurses jedoch nicht behandelt.

Storage

Dieses Menü enthält Optionen zum Konfigurieren des persistenten Storages für Projektanwendungen. Insbesondere werden persistente Volumes und Anforderungen für ein persistentes Volume für ein Projekt über das Menü „menu:Storage“ verwaltet.

Builds

Die Option **Build Configs** zeigt eine Liste der Projekt-Build-Konfigurationen an. Klicken Sie auf einen Build-Konfigurationslink in dieser Ansicht, um auf eine Übersichtsseite für die angegebene Build-Konfiguration zuzugreifen. Auf dieser Seite können Sie die Build-Konfiguration der Anwendung anzeigen und bearbeiten.

Die Option **Builds** stellt eine Liste der letzten Erstellungsprozesse für Anwendungscontainer-Images im Projekt bereit. Klicken Sie auf den Link für einen bestimmten Build, um auf die Build-Protokolle für diesen bestimmten Build-Prozess zuzugreifen.

Die Option **Image Streams** stellt eine Liste der im Projekt definierten Image-Streams bereit. Klicken Sie auf einen Image-Stream-Eintrag in dieser Liste, um auf eine Übersichtsseite zuzugreifen, um diesen Image-Stream anzuzeigen und zu verwalten.

Compute

Bietet Optionen zum Zugriff und zur Verwaltung der Server-Knoten des OpenShift-Clusters. Auf dieser Seite können Sie auch Knoten-Integritätsprüfungen und die automatische Skalierung konfigurieren.

User management

Mit dieser Option können Sie die Authentifizierung und Autorisierung mit Benutzern, Gruppen, Rollen, Rollenbindungen und Servicekonten konfigurieren.

Administration

Bietet Optionen zum Verwalten von Cluster- und Projekteinstellungen, z. B. Ressourcenkontingente und rollenbasierte Zugriffskontrollen. Die Funktionen im Abschnitt Administration werden in diesem Kurs nicht erläutert.

Erstellen neuer Anwendungen

Verwenden Sie **+ Add** in der Perspektive **Developer**, um eine Möglichkeit zum Erstellen einer neuen Anwendung in einem OpenShift-Projekt auszuwählen. Sie können eine Anwendung aus dem **Developer Catalog** hinzufügen, die zum Erstellen technologiespezifischer Anwendungen eine Auswahl von Source-to-Image(S2I)-Vorlagen, Builder-Images und Helm-Diagrammen bietet. Wählen Sie eine gewünschte Vorlage aus, und geben Sie die erforderlichen Informationen für die Bereitstellung der neuen Anwendung an.

Sie sind nicht darauf beschränkt, eine Anwendung aus dem Katalog bereitzustellen. Sie können eine Anwendung auch wie folgt bereitstellen:

- Mit einem Container-Image, das in einer Remote-Container-Registry gehostet wird.

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

- Mit einer YAML-Datei, welche die zu erstellenden Kubernetes- und OpenShift-Ressourcen angibt.
- Mit einem Builder-Image unter Verwendung des Quellcodes aus Ihrem Git-Repository.
- Ein Dockerfile.

Abbildung 6.10: OpenShift Developer Catalog-Seite

Um eine Anwendung mit einer dieser Methoden zu erstellen, wählen Sie die entsprechende Option auf der Seite **Add** aus. Verwenden Sie die Option **Container Image**, um ein vorhandenes Container-Image bereitzustellen. Verwenden Sie die Option **YAML**, um die in einer YAML-Datei angegebenen Ressourcen zu erstellen. Verwenden Sie die Option **From Git**, um den Quellcode mithilfe eines Builder-Images bereitzustellen. Verwenden Sie die Option **From Dockerfile**, um das Image aus einem bestimmten Dockerfile anzugeben und zu erstellen. Die Optionen **Databases**, **Operator Backed** und **Helm Chart** sind Verknüpfungen zum Katalog.

Verwalten von Anwendungs-Builds

Klicken Sie in der Perspektive **Administrator** im Menü „menu:Builds“ auf die Option **Build Configs**, nachdem Sie einem Projekt eine Source-to-Image-Anwendung hinzugefügt haben. Über diese Ansicht kann auf die neue Build-Konfiguration zugegriffen werden:

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

Name	Namespace	Labels	Created
BC todoapp	NS test	app=todoapp	Aug 3, 1:42 pm

Abbildung 6.11: „Build Configs“-Seite in OpenShift

Klicken Sie auf eine Build-Konfiguration in der Liste, um eine Übersichtsseite für die ausgewählte Build-Konfiguration anzuzeigen. Auf der Übersichtsseite können Sie:

- Die Parameter der Build-Konfiguration anzeigen, beispielsweise die URL für das Git-Repository des Quellcodes.
- Während eines Anwendungs-Build-Prozesses im Builder-Container festgelegte Umgebungsvariablen anzeigen und bearbeiten.
- Eine Liste der aktuellen Anwendungs-Builds anzeigen und auf einen ausgewählten Build klicken, um auf die Build-Prozessprotokolle zuzugreifen.

Verwalten von bereitgestellten Anwendungen

Über das Menü „Workloads“ kann auf die Bereitstellungen im Projekt zugegriffen werden:

Name	Namespace	Status	Labels	Pod Selector
DC todoapp	NS test	1 of 1 pods	app=todoapp	app=todoapp, deploymentconfig=todoapp

Abbildung 6.12: OpenShift-Workloads-Menü

Unter „Workloads“ finden Sie viele der im Kurs behandelten Konstrukte, darunter Pods, Bereitstellungen und Konfigurations-Maps.

Klicken Sie auf einen Bereitstellungseintrag in der Liste, um eine Übersichtsseite für die Auswahl anzuzeigen. Auf der Übersichtsseite können Sie:

- Die Parameter der Bereitstellungen anzeigen, beispielsweise die Spezifikationen eines Anwendungs-Container-Images.
- Die gewünschte Anzahl der Anwendungs-Pods ändern, um die Anwendung manuell zu skalieren.
- Anzeigen und Bearbeiten der Umgebungsvariablen, die im bereitgestellten Anwendungs-Container festgelegt sind.
- Eine Liste der Anwendungs-Pods anzeigen und auf einen ausgewählten Pod klicken, um auf die Protokolle für diesen Pod zuzugreifen.

Weitere Funktionen der Web Console

Die Web Console ermöglicht Folgendes:

- Ressourcen wie Projektcontingente, Benutzermitgliedschaften, Geheimnisse und andere erweiterte Ressourcen verwalten
- Anforderungen für persistente Volumes erstellen
- Builds, Pods, Bereitstellungen und Systemereignisse überwachen
- Erstellen von CI(Continuous Integration)- und CD(Continuous Deployment)-Pipelines mit Jenkins.

Die detaillierte Verwendung der oben genannten Features liegt nicht im Rahmen dieses Kurses.

► Angeleitete Übung

Erstellen von Anwendungen mit der Web Console

In dieser Übung erstellen Sie eine Anwendung in einem OpenShift-Cluster mithilfe der OpenShift Web Console und stellen diese bereit.

Ergebnisse

Sie sollten in der Lage sein, eine Anwendung in einem OpenShift-Cluster mithilfe der Web Console zu erstellen und bereitzustellen.

Bevor Sie Beginnen

Rufen Sie die Lab-Dateien ab, indem Sie das Lab-Skript ausführen:

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

Das Lab-Skript verifiziert, dass der OpenShift-Cluster ausgeführt wird.

Anweisungen

- 1. Überprüfen Sie den PHP-Quellcode für die Beispielanwendung, und erstellen und übertragen Sie einen neuen Branch namens `console`, der während dieser Übung verwendet werden soll.
- 1.1. Wechseln Sie zu Ihrem lokalen Klon des Git-Repositorys `D0180-apps`, und checken Sie den Branch `master` des Kurs-Repositorys aus, um sicherzustellen, dass Sie diese Übung von einem als fehlerfrei bekannten Zustand aus starten:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 1.2. Erstellen Sie einen neuen Branch, um alle Änderungen zu speichern, die Sie während dieser Übung vornehmen:

```
[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. Überprüfen Sie den PHP-Quellcode der Anwendung im Ordner `php-helloworld`. Öffnen Sie die Datei `index.php` im Ordner `/home/student/D0180-apps/php-helloworld`:

```
<?php  
print "Hello, World! php version is " . PHP_VERSION . "\n";  
?>
```

Die Anwendung implementiert eine einfache Antwort, die die ausgeführte PHP-Version zurückgibt.

- 2. Öffnen Sie einen Webbrowser, und navigieren Sie zu `https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}`, um auf die OpenShift Web Console zuzugreifen. Melden Sie sich an, und erstellen Sie ein neues Projekt mit dem Namen `youruser-console`.

- 2.1. Laden Sie die Konfiguration Ihrer Kursumgebung.

Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

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

- 2.2. Rufen Sie den Wert der für Ihr Cluster spezifischen Platzhalter-Domain mit `$RHT_OCP4_WILDCARD_DOMAIN` ab.

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

- 2.3. Öffnen Sie Firefox-Browser, und navigieren Sie zu `https://console-openshift-console.${RHT_OCP4_WILDCARD_DOMAIN}`, um auf die OpenShift Web Console zuzugreifen. Melden Sie sich mit Ihren Anmeldedaten bei der OpenShift Web Console an.
- 2.4. Erstellen Sie ein neues Projekt mit dem Namen `youruser-console`. Sie können in die anderen Felder beliebige Werte eingeben.

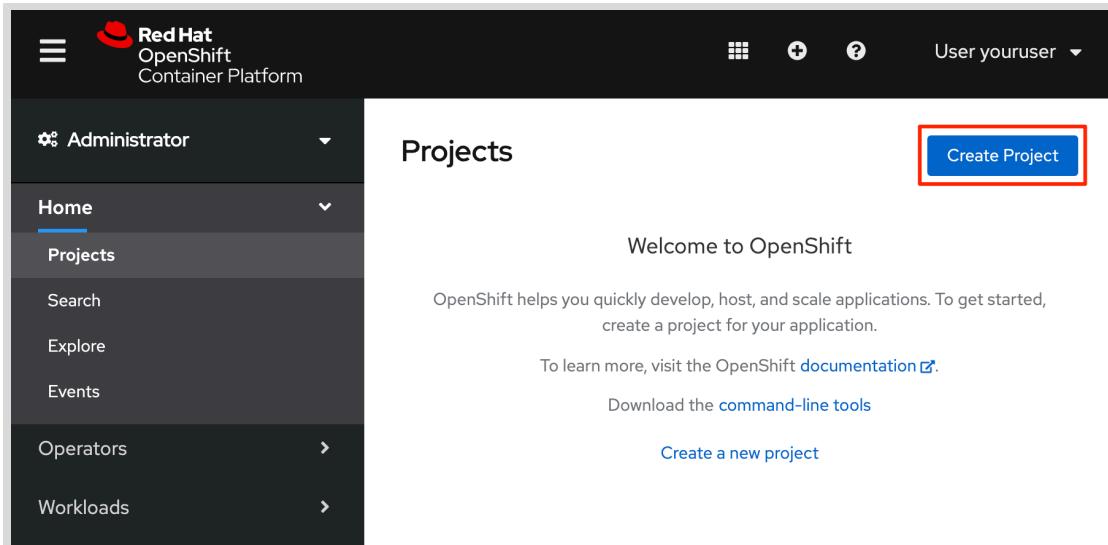


Abbildung 6.13: Neues Projekt erstellen

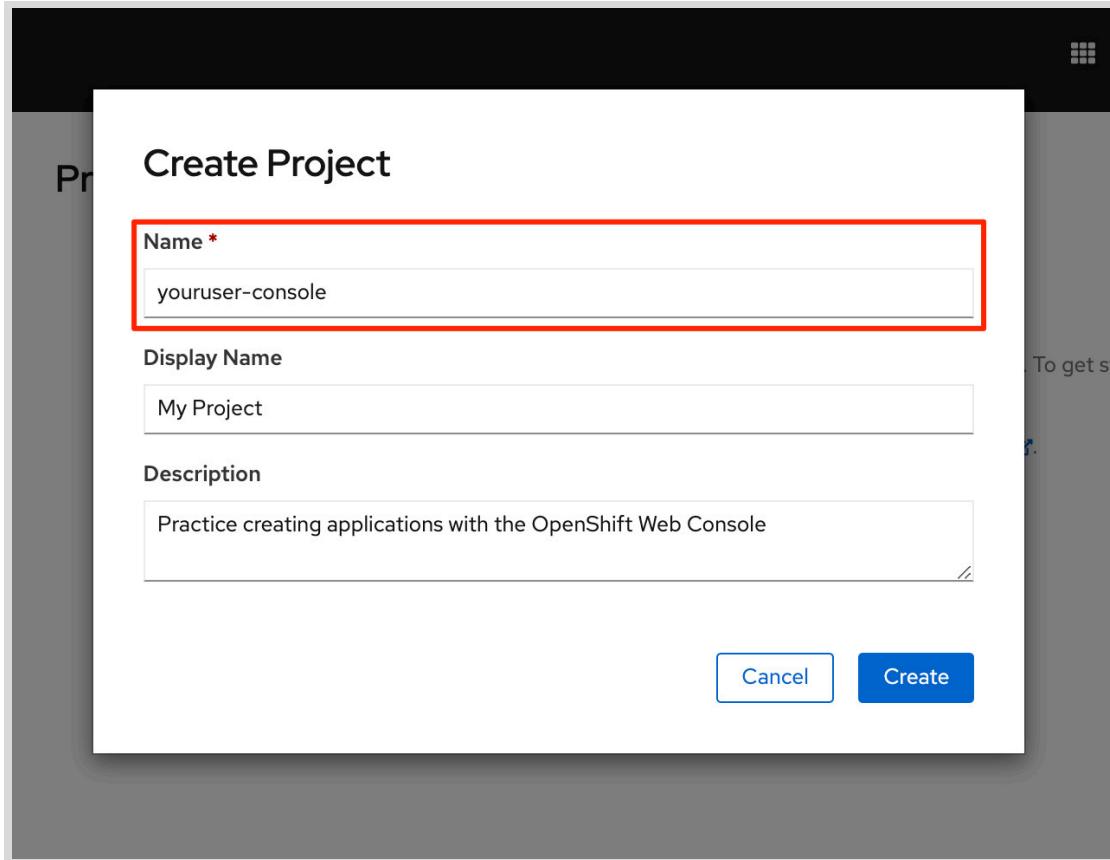


Abbildung 6.14: Neues Projekt erstellen

- 2.5. Klicken Sie nach dem Ausfüllen der Pflichtfelder im Dialogfeld **Create Project** auf **Create**, um zur Seite **Project Status** für das Projekt **youruser-console** zu gelangen:

Details		Status		Activity	
Name	youruser-console	Active		Ongoing	
Requester	youruser			There are no ongoing activities.	
Labels	No labels	Utilization		Recent Events	
		Resource	Usage	Pause	
Inventory		CPU	Not available	There are no recent events.	
0 Deployments		Memory	Not available		
0 Deployment Configs		Filesystem	Not available		
0 Stateful Sets					

Abbildung 6.15: Seite „Project Status“

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

► 3. Erstellen Sie die neue Anwendung `php-helloworld` mithilfe einer PHP-Vorlage.

3.1. Wählen Sie in der Liste oben im linken Menü die Perspektive **Developer** aus.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar has a dropdown menu with options: Administrator, Administrator, Developer (which is highlighted with a red box), Search, Explore, Events, Operators, and Workloads. The main content area shows a project named 'PR youruser-console' which is Active. The 'Overview' tab is selected. It displays the project's name ('youruser-console') and status ('Active'). Below the status, there are sections for 'Requester' and 'Details'.

Abbildung 6.16: Dropdown-Menü für die Perspektive „Developer“

3.2. Klicken Sie auf **From Catalog**, um eine Liste der Technologievorlagen anzuzeigen. Deaktivieren Sie das Kontrollkästchen **Operator Backed**, um alle Vorlagen anzuzeigen.

The screenshot shows the 'Developer Catalog' page. The left sidebar includes options like +Add, Topology, Monitoring, Search, Builds, Helm, Project, Config Maps, and Secrets. The main area is titled 'Developer Catalog' and contains a sub-header: 'Add shared apps, services, or source-to-image builders to your project from the Developer Catalog. Cluster admins can install additional shared apps from the catalog to show up here automatically.' Below this, there are two tabs: 'All Items' (selected) and 'All Items'. There is a search bar 'Filter by keyword...' and a 'Group By: None' dropdown. On the left, there is a sidebar with categories: Languages, Databases, Middleware, CI/CD, and Other. Under 'Type', there are checkboxes for Operator Backed (0), Helm Charts (8), Builder Image (10), Template (98), and Service Class (0). The main content area displays four catalog items:

- .NET Core** (Builder Image): .NET Core. Build and run .NET Core 3.1 applications on RHEL 7. For more information about using this...
- .NET Core** (Template): .NET Core + PostgreSQL (Persistent). An example .NET Core application with a PostgreSQL database. For more information about using th...
- .NET Core** (Template): .NET Core Example. An example .NET Core application.
- 3scale APIcast API Gateway** (Template): 3scale APIcast API Gateway provided by Red Hat, Inc. 3scale's APIcast is an NGINX based API gateway used to...

Abbildung 6.17: Developer Catalog-Seite

3.3. Geben Sie `php` im Feld **Filter by keyword** ein.

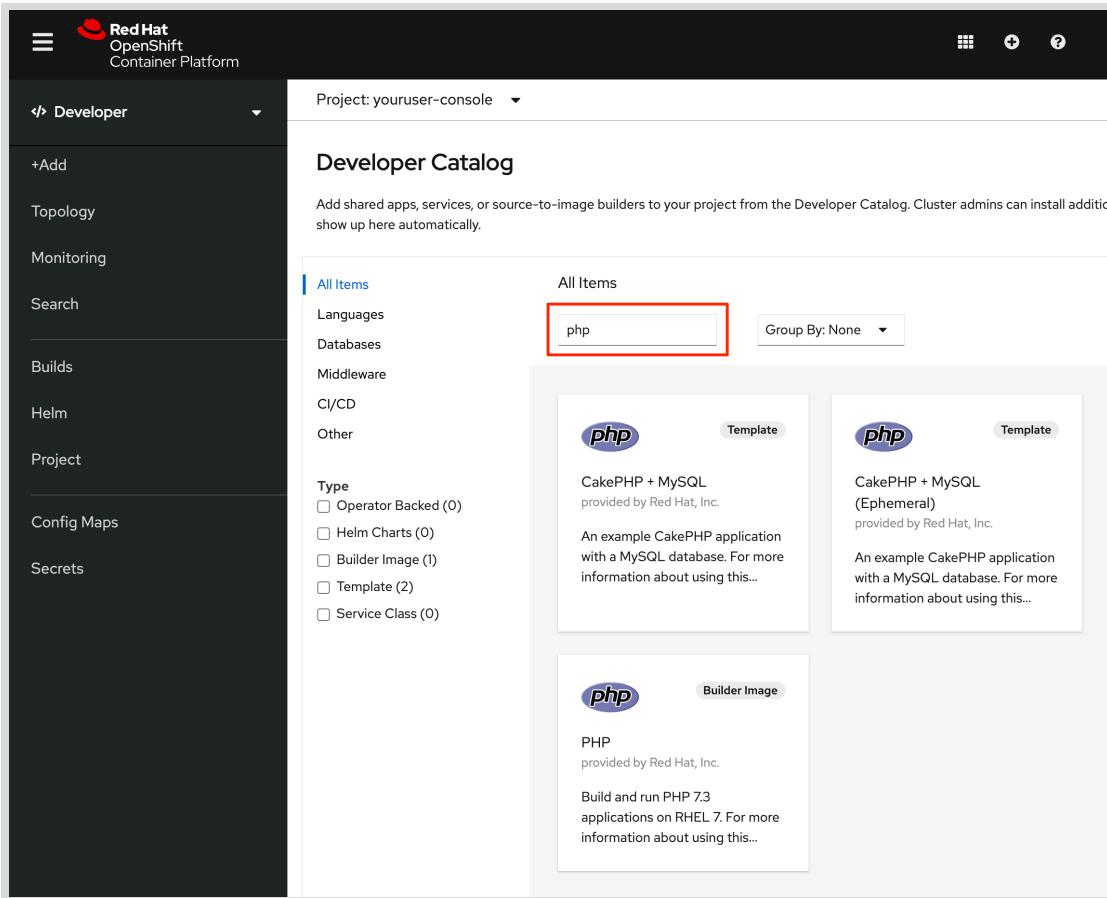


Abbildung 6.18: Suchen nach PHP-Vorlagen

- 3.4. Klicken Sie nach dem Filtern auf das PHP-Builder-Image, um das PHP-Dialogfeld anzuzeigen. Klicken Sie auf **Create Application**, um die Seite **Create Source-to-Image Application** anzuzeigen.

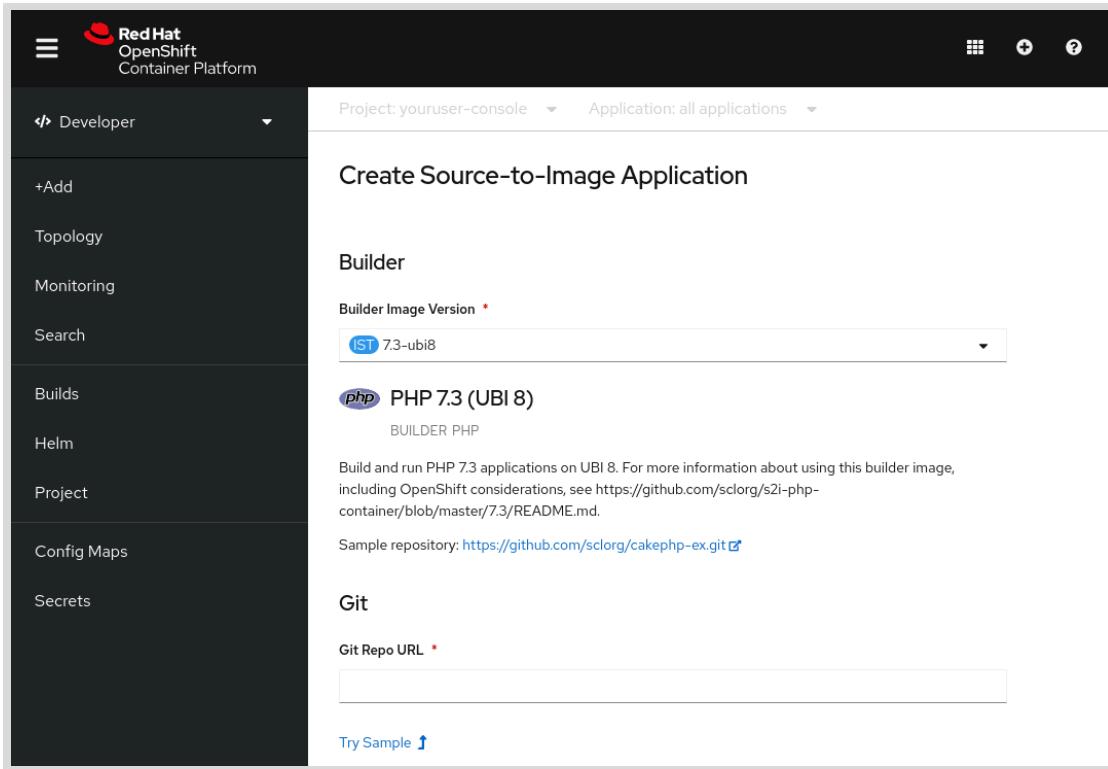


Abbildung 6.19: Konfigurieren von Source-to-Image für eine PHP-Anwendung

- 3.5. Ändern Sie die **Builder Image Version** in die PHP-Version 7.3 (UBI 8).
Geben Sie den Speicherort des Quellcode-Git-Repositorys an: `https://github.com/yourgituser/D0180-apps`
Verwenden Sie **Advanced Git Options**, um das Kontextverzeichnis auf `php-helloworld` festzulegen und `console` für diese Übung zu verzweigen.

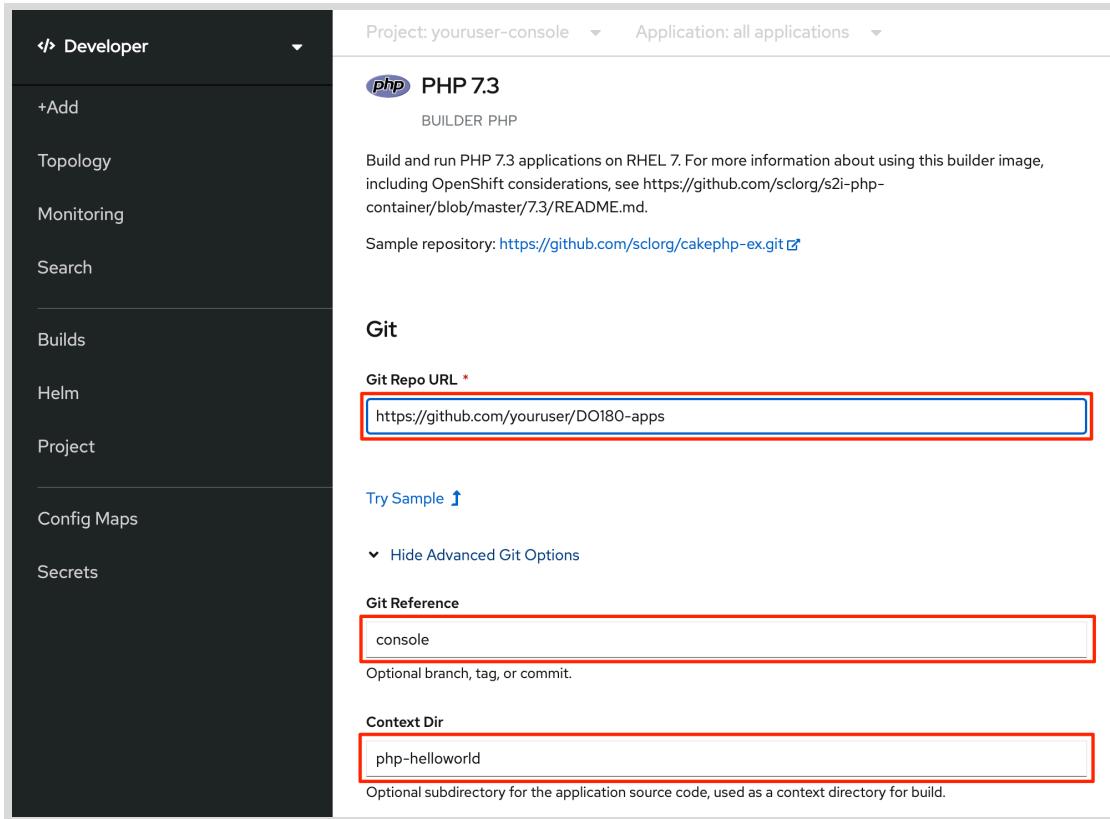


Abbildung 6.20: Festlegen erweiterter Git-Optionen für die Anwendung

Geben Sie php-helloworld für den Anwendungsnamen und den Namen ein, der für die zugeordneten Ressourcen verwendet wird. Wählen Sie **Deployment** als Ressourcentyp aus.

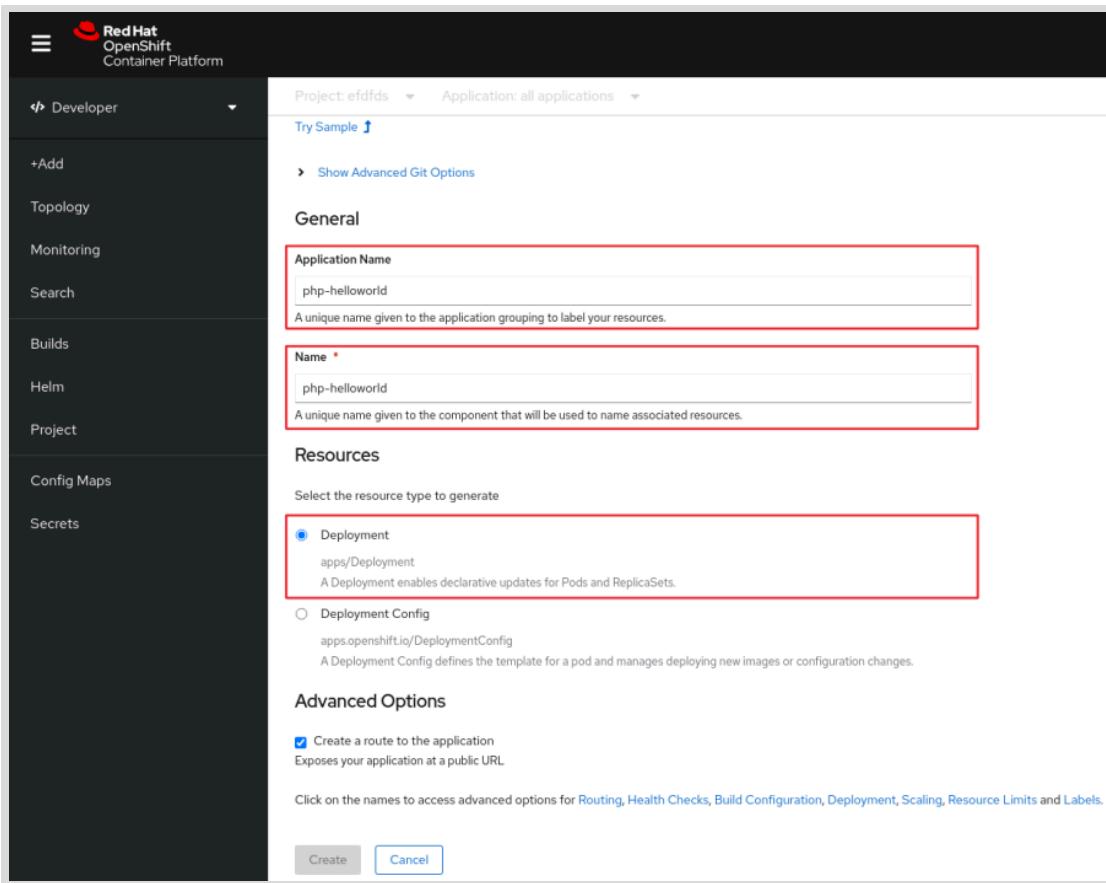
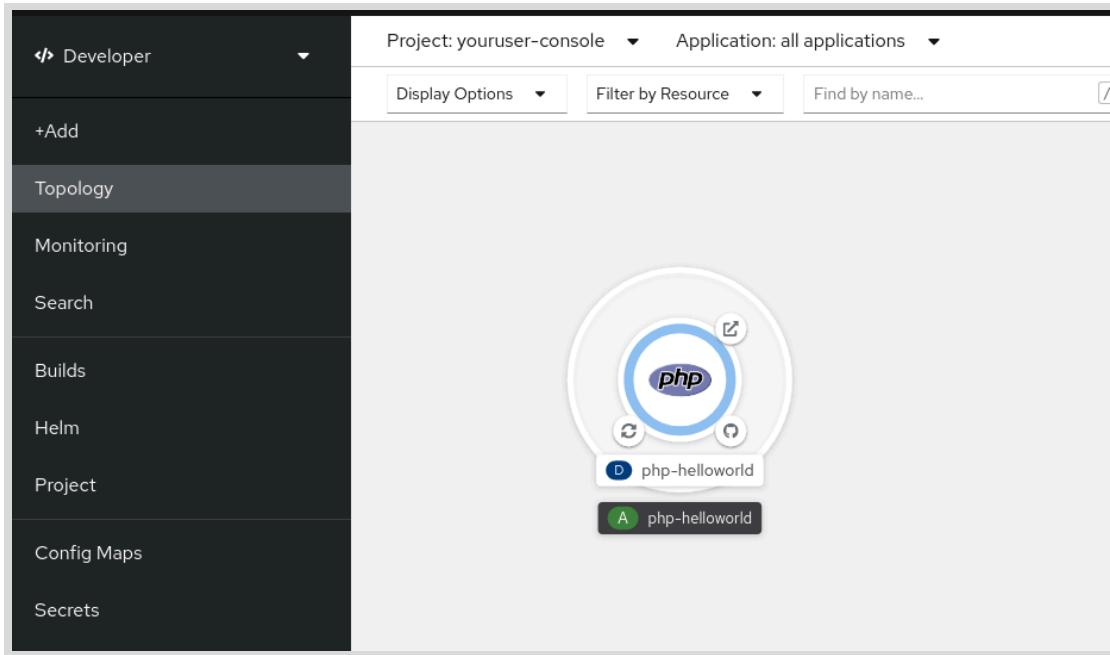


Abbildung 6.21: Festlegen von Anwendungsoptionen

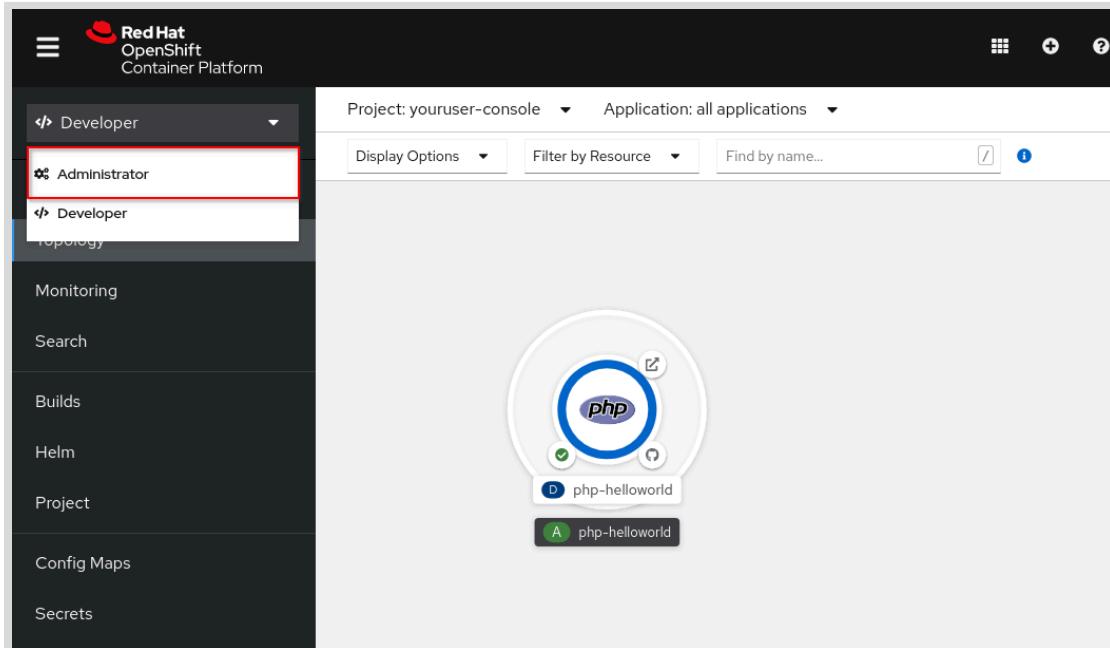
Scrollen Sie zum Ende der Seite, und aktivieren Sie **Create a route to the application**. Klicken Sie auf **Create**, um die erforderlichen OpenShift- und Kubernetes-Ressourcen für die Anwendung zu erstellen.

Sie werden zur Seite **Topology** umgeleitet:

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift**Abbildung 6.22: Seite „Topology“**

Auf dieser Seite wird angezeigt, dass die Anwendung `php-helloworld` erstellt wird. Die Anmerkung D auf der linken Seite des Links `php-helloworld` ist ein Akronym für Deployment. Über diesen Link gelangen Sie zu einer Seite, die Informationen zur Bereitstellung der Anwendung enthält.

3.6. Wechseln Sie für den Rest der Übung zurück zur Perspektive **Administrator**:

**Abbildung 6.23: Dropdown-Menü mit Administrator-Perspektive**

- 4. Verwenden Sie die Navigationsleiste auf der linken Seite der OpenShift Web Console, um Informationen zu den OpenShift- und Kubernetes-Ressourcen der Anwendung zu finden:

- Bereitstellungen
 - BuildConfig
 - Build-Protokolle
 - Service
 - Route
- 4.1. Untersuchen Sie die Bereitstellung. Klicken Sie auf der Navigationsleiste auf **Workloads**, um weitere Menüoptionen anzuzeigen. Klicken Sie auf „Deployments“, um eine Liste der Bereitstellungen für das Projekt *youruser-console* anzuzeigen. Klicken Sie auf den Link [php-helloworld](#), um Informationen zur Bereitstellung anzuzeigen.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is collapsed. The main navigation bar at the top has the Red Hat logo and the text "Red Hat OpenShift Container Platform". Below the navigation bar, the user is logged in as "Administrator". The left sidebar menu includes "Home", "Projects", "Search", "Explore", "Events", "Operators", "Workloads" (which is expanded to show "Pods", "Deployments" - which is selected and highlighted in blue, "Deployment Configs", "Stateful Sets", "Secrets", "Config Maps", "Cron Jobs", "Jobs", "Daemon Sets", "Replica Sets", and "Replication Controllers"). The main content area shows the "Deployment Details" for the "php-helloworld" deployment under the "youruser-console" project. The "Details" tab is selected. At the top of the details page, there is a large circular icon containing the number "1 pod". Below this, the deployment is identified as "Name: php-helloworld" and "Namespace: NS youruser-console". The "Update Strategy" is set to "RollingUpdate". The "Max Unavailable" is "25% of 1 pod". The "Max Surge" is "25% greater than 1 pod". The "Progress Deadline Seconds" is "600 seconds". The "Min Ready Seconds" is "Not Configured". The "Labels" section lists several key labels: app=php-helloworld, app.kubernetes.io/component=php-helloworld, app.kubernetes.io/instance=php-helloworld, app.kubernetes.io/name=php, app.kubernetes.io/part-of=php-helloworld, app.openshift.io/runtime=php, and app.openshift.io/runtime-version=7.3-ubi8. A "Pod Selector" search bar contains the query "app=php-helloworld".

Abbildung 6.24: Seite mit den Details zur Anwendungsbereitstellung

Lesen Sie auf der Registerkarte **Details** die verfügbaren Informationen. Der Build wird möglicherweise noch ausgeführt, wenn Sie auf diese Seite zugreifen. Daher weist „Deployment“ möglicherweise noch nicht den Wert **1 pod** auf.

Wenn Sie neben dem Ringdiagramm, das die Anzahl der Pods angibt, auf die Aufwärts- und Abwärtspfeile klicken, können Sie die Anwendung horizontal hoch- und herunterskalieren.

- 4.2. Untersuchen Sie die Build-Konfiguration. Klicken Sie auf der Navigationsleiste auf **Builds**, um weitere Menüoptionen anzuzeigen. Klicken Sie auf „> Build Configs“, um eine Liste der Build-Konfigurationen für das Projekt *youruser-console* anzuzeigen. Klicken Sie auf den Link [php-helloworld](#), um die Build-Konfiguration für die Anwendung anzuzeigen.

The screenshot shows the Red Hat OpenShift Container Platform web interface. The left sidebar is a navigation menu with sections like Home, Projects, Search, Explore, Events, Operators, Workloads, Networking, Storage, Builds, Build Configs, Builds, Image Streams, Compute, User Management, and Administration. The 'Builds' section is currently selected. The main content area is titled 'Build Config Details' for a build config named 'php-helloworld'. It shows the following details:

- Name:** php-helloworld (Type: Source)
- Namespace:** NS youruser-console (Git Repository: https://github.com/yourgithubuser/DO180-apps.git)
- Labels:** app:php-helloworld, app.kubernetes.io/component:php-helloworld, app.kubernetes.io/instance:php-helloworld, app.kubernetes.io/name:php, app.kubernetes.io/part-of:console, app.openshift.io/runtime:php, app.openshift.io/runtime-version:7.3 (Git Ref: console, Context Dir: php-helloworld)
- Annotations:** 3 Annotations (Build From: IS^T php:7.3, Output To: IS^T php-helloworld:latest)
- Created At:** 4 minutes ago (Run Policy: Serial)
- Owner:** No owner

Abbildung 6.25: Seite mit den Konfigurationsdetails eines Anwendungs-Builds

Lesen Sie auf der Registerkarte **Details** die verfügbaren Informationen. Auf der Registerkarte **YAML** können Sie die Build-Konfiguration als eine YAML-Datei anzeigen und bearbeiten. Die Registerkarte **Builds** enthält eine Verlaufsliste der Builds sowie einen Link zu weiteren Informationen zu jedem Build. Auf der Registerkarte **Environment** können Sie Umgebungsvariablen für die Build-Umgebung der Anwendung anzeigen und bearbeiten. Auf der Registerkarte **Events** wird eine Liste mit Build-bezogenen Ereignissen und Metadaten angezeigt.

- 4.3. Untersuchen Sie die Protokolle für den Source-to-Image Build der Anwendung. Klicken Sie im Menü **Builds** auf „Builds“, um eine Liste der neuesten Builds für das Projekt *youruser-console* anzuzeigen. Klicken Sie auf den Link [php-helloworld-1](#), um auf die Informationen für den ersten Build der Anwendung *php-helloworld* zuzugreifen:

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

The screenshot shows the Red Hat OpenShift Container Platform web interface. On the left, a dark sidebar menu includes options like Home, Projects, Search, Explore, Events, Operators, Workloads, Networking, Storage, Builds (selected), Build Configs, Builds (highlighted), Image Streams, Compute, User Management, and Administration. The main content area is titled 'Build Details' for a build named 'php-helloworld-1' (status: Complete). It features three resource usage charts: Memory Usage (100 MB), CPU Usage (100m), and Filesystem (0 B). Below the charts, detailed information is provided: Name (php-helloworld-1), Status (Complete), Namespace (youruser-console), Type (Source), Labels (app=php-helloworld, app.kubernetes.io/part-of=console, app.kubernetes.io/instance=php-helloworld), and Git Repository (https://github.com/yourgithubuser/DO180-apps.git).

Abbildung 6.26: Seite mit den Details eines Anwendungs-Builds

Lesen Sie auf der Registerkarte **Details** die verfügbaren Informationen. Klicken Sie anschließend auf die Registerkarte **Logs**. Ein scrollbares Textfeld enthält die Ausgabe des Erstellungsprozesses:

This screenshot shows the same Red Hat OpenShift interface as above, but the 'Logs' tab is selected under the 'Build Details' header. The log output is displayed in a scrollable text area. The logs show the build process starting with 'Log stream ended.' followed by 59 lines of command-line output. The output details the creation of a Docker image, pushing it to an image registry, and successfully pushing the image to port 5000. Key log entries include:

```

Log stream ended.
59 lines
Writing manifest to image destination
Storing signatures
--> 5cb438a2c45
5cb438a2c45f80b7620779e8ca5b10b9c7e8f3279718686d067dd45fd9dec5de
Pushing image image-registry.openshift-image-registry.svc:5000/youruser-console/php-helloworld:latest ...
Getting image source signatures
Copying blob sha256:65b33f0244f1283afca60b050b38243d16fba6725550a6abff9eb5b67ba59b461
Copying blob sha256:fcc5b206e9329a1674dd9e8efbee45c9be28000dcbabba3c6bb67a2f2ccfc2a
Copying blob sha256:9f04ed9c572e5de5b715b5f831e573d1ad7080933982a25113c443eb6bd9497e
Copying blob sha256:c9ae496e61018b89a5d0bd37ef429b5a893e52cff040a40e325be5117620c1
Copying blob sha256:e7021e0589e97471d99c4265b7e864da328e48f116b5f2f60353b2e0a2adb373
Copying blob sha256:9e7a6dc796f0a75c560158a9f9e30fb5a90cb53edce9fbfd5778406e4de39
Copying config sha256:5cb438a2c45f80b7620779e8ca5b10b9c7e8f3279718686d067dd45fd9dec5de
Writing manifest to image destination
Storing signatures
Successfully pushed image-registry.openshift-image-registry.svc:5000/youruser-console/php-helloworld@sha256:5cb438a2c45f80b7620779e8ca5b10b9c7e8f3279718686d067dd45fd9dec5de
Push successful

```

Abbildung 6.27: Protokolle für einen Anwendungs-Build

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

Wenn Podman ein Container-Image erstellt, wird eine ähnliche Ausgabe verglichen mit der im Browser angezeigten Ausgabe beobachtet.

- 4.4. Suchen Sie nach Informationen für den Service der Anwendung `php-helloworld`. Klicken Sie auf der Navigationsleiste auf **Networking**, um weitere Menüoptionen anzuzeigen. Klicken Sie auf „> Services“, um eine Liste der Services für das Projekt `youruser-console` anzuzeigen. Klicken Sie auf den Link `php-helloworld`, um die dem Service der Anwendung zugeordneten Informationen anzuzeigen:

Name	Type	Port	Protocol	Pod Port or Name
8080-tcp	S	8080	TCP	P 8080
8443-tcp	S	8443	TCP	P 8443

Abbildung 6.28: Seite mit Servicedetails

Lesen Sie auf der Registerkarte **Details** die verfügbaren Informationen. Auf der Registerkarte **YAML** können Sie die Servicekonfiguration als eine YAML-Datei anzeigen und bearbeiten. Auf der Registerkarte **Pods** wird die aktuelle Liste der Pods angezeigt, die den Anwendungsservice bereitstellen.

- 4.5. Suchen Sie nach externen Routeninformationen für die Anwendung. Klicken Sie in der Navigationsleiste auf **Networking** → **Routes**, um eine Liste der konfigurierten Routen für das Projekt `youruser-console` anzuzeigen. Klicken Sie auf den Link `php-helloworld`, um die der Route der Anwendung zugeordneten Informationen anzuzeigen:

Abbildung 6.29: Seite mit Routendetails

Lesen Sie auf der Registerkarte **Details** die verfügbaren Informationen. Das Feld **Location** enthält einen Link zur externen Route für die Anwendung: `http://php-helloworld-${{RHT_OCP4_DEV_USER}}-console`. `-${{RHT_OCP4_WILDCARD_DOMAIN}}`. Klicken Sie auf den Link, um auf die Anwendung in einer neuen Registerkarte zuzugreifen:

Abbildung 6.30: Erste PHP-Anwendungsergebnisse

- 5. Ändern Sie den Anwendungscode, übernehmen Sie die Änderung, senden Sie den Code an das externe Git-Repository und lösen Sie einen neuen Anwendungs-Build aus.

- 5.1. Geben Sie das Quellcodeverzeichnis ein:

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

- 5.2. Fügen Sie die zweite Druckzeilenanweisung auf der Seite `index.php` hinzu, damit die Nachricht „A change is in the air!“ angezeigt wird. Speichern Sie die Datei. Fügen Sie die Änderung zum Git-Index hinzu, übernehmen Sie die Änderung und senden Sie die Änderungen an das externe Git-Repository.

```
[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. Lösen Sie manuell über die Web Console ein Anwendungs-Build aus.

Klicken Sie in der Navigationsleiste auf **Builds** → **Build Configs** und dann auf den Link **php-helloworld**, um auf die Seite „Build Config Details“ zuzugreifen. Klicken Sie im Menü **Actions** in der oberen rechten Ecke des Bildschirms auf **Start Build**:

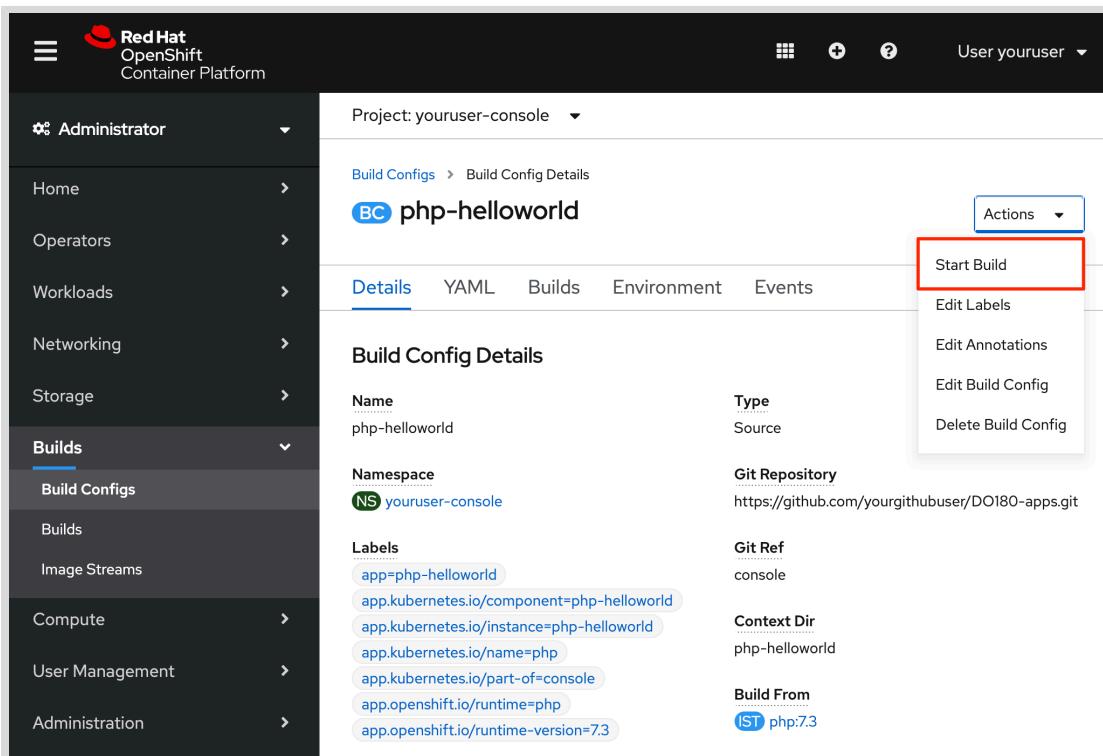


Abbildung 6.31: Starten eines Anwendungs-Builds

Sie werden auf eine Seite mit den Build-Details für den neuen Build umgeleitet. Klicken Sie auf die Registerkarte **Logs**, um den Fortschritt des Builds zu überwachen. Die letzte Zeile eines erfolgreichen Builds enthält **Push successful**.

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

Nach Abschluss des Builds wird die Bereitstellung gestartet. Navigieren Sie zum Abschnitt **Workloads** → **Pods**, und warten Sie auf die Bereitstellung und Ausführung des neuen Pods.

- 5.4. Laden Sie die URL `http://php-helloworld-${RHT_OCP4_DEV_USER}-console.${RHT_OCP4_WILDCARD_DOMAIN}` im Browser neu. Die Anwendungsantwort entspricht dem aktualisierten Quellcode:



Abbildung 6.32: Aktualisierte Webanwendungsausgabe

- ▶ 6. Löschen Sie das Projekt. Klicken Sie auf der Navigationsleiste auf **Home** → **Projects**. Klicken Sie auf das Symbol rechts neben der Zeile, die einen Eintrag für das Projekt `youruser-console` enthält. Klicken Sie im angezeigten Menü auf „> Delete Project“.

The screenshot shows the Red Hat OpenShift web console interface. The left sidebar is open, showing 'Administrator' and a list of navigation items: Home (selected), Projects, Search, Explore, Events, Operators, and Workloads. The main content area is titled 'Projects' and contains a table with one row. The row has columns: Name (PR youruser-console), Display Name (My Project), Status (Active), Requester (youruser), and a more options menu. A red box highlights the 'Delete Project' button in the bottom right corner of the row's details panel.

Abbildung 6.33: Löschen eines Projekts

Geben Sie im Bestätigungsdialogfeld `youruser-console` ein, und klicken Sie auf **Delete**.

Beenden

Führen Sie auf `workstation` das Skript `lab openshift-webconsole finish` aus, um diese praktische Übung abzuschließen.

```
[student@workstation php-helloworld]$ lab openshift-webconsole finish
```

Hiermit ist die angeleitete Übung beendet.

► Praktische Übung

Bereitstellen containerisierter Anwendungen in OpenShift

Ergebnisse

Sie sollten in der Lage sein, eine OpenShift-Anwendung zu erstellen und über einen Webbrowser darauf zuzugreifen.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Bereiten Sie die Umgebung für die praktische Übung vor.
2. Erstellen Sie eine Anwendung zur Temperaturumrechnung in PHP unter Verwendung des Image-Stream-Tags `php:7.3`. Der Quellcode ist im Git-Repository im Verzeichnis `temps` unter <https://github.com/RedHatTraining/D0180-apps/> verfügbar. Sie können die Anwendung über die Befehlszeilenschnittstelle von OpenShift oder die Web Console erstellen.
3. Verifizieren Sie, dass Sie in einem Webbrowser unter `http://temps- ${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN}` auf die Anwendung zugreifen können.

Bewertung

Führen Sie auf `workstation` den Befehl `lab openshift-review grade` aus, um Ihre Arbeit zu bewerten. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf `workstation` den Befehl `lab openshift-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Bereitstellen containerisierter Anwendungen in OpenShift

Ergebnisse

Sie sollten in der Lage sein, eine OpenShift-Anwendung zu erstellen und über einen Webbrowser darauf zuzugreifen.

Bevor Sie Beginnen

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Bereiten Sie die Umgebung für die praktische Übung vor.
 - 1.1. Laden Sie die Konfiguration Ihrer Kursumgebung.
Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

```
[student@workstation ~]$ source /usr/local/etc/ocp4.config
```
 - 1.2. Melden Sie sich beim OpenShift-Cluster an.

```
[student@workstation ~]$ oc login -u ${RHT_OCP4_DEV_USER} -p \
> ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful
...output omitted...
```
- 1.3. Erstellen Sie ein neues Projekt namens „\${RHT_OCP4_DEV_USER}-ocp“ für die Ressourcen, die Sie während dieser Übung erstellen:

```
[student@workstation ~]$ oc new-project ${RHT_OCP4_DEV_USER}-ocp
```

2. Erstellen Sie eine Anwendung zur Temperaturumrechnung in PHP unter Verwendung des Image-Stream-Tags `php:7.3`. Der Quellcode ist im Git-Repository im Verzeichnis `temps` unter <https://github.com/RedHatTraining/D0180-apps/> verfügbar. Sie können die Anwendung über die Befehlszeilenschnittstelle von OpenShift oder die Web Console erstellen.
 - 2.1. Führen Sie an der Befehlszeilenschnittstelle die folgenden Befehle aus:

Kapitel 6 | Bereitstellen containerisierter Anwendungen in OpenShift

```
[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. Überwachen Sie den Fortschritt des Builds.

```
[student@workstation ~]$ oc logs -f bc/temps
Cloning "https://github.com/RedHatTraining/D0180-apps" ...
Commit: f7cd8963ef353d9173c3a21dccc402f3616840b (Initial commit, including all
apps previously in course)
...output omitted...
Successfully pushed image-registry.openshift-image-registry.svc:5000/
${RHT_OCP4_DEV_USER}-temps
Push successful
```

2.3. Verifizieren Sie, dass die Anwendung bereitgestellt wurde.

```
[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
```

Drücken Sie Strg+C, um den Befehl `oc get pods -w` zu beenden.

2.4. Stellen Sie den Service `temps` bereit, um eine externe Route für die Anwendung zu erstellen.

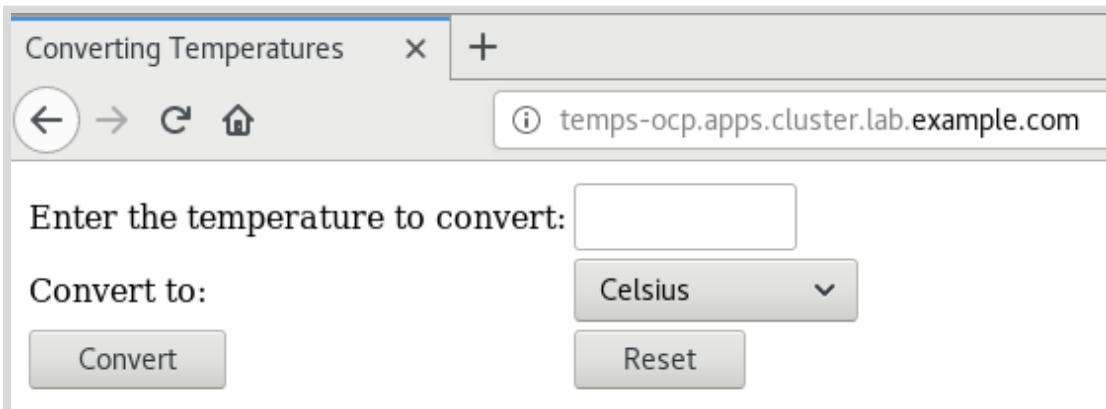
```
[student@workstation ~]$ oc expose svc/temps
route.route.openshift.io/temps exposed
```

3. Verifizieren Sie, dass Sie in einem Webbrowser unter `http://temps- ${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN}` auf die Anwendung zugreifen können.

- 3.1. Ermitteln Sie die URL für die Route.

```
[student@workstation ~]$ oc get route/temps
NAME      HOST/PORT
temps     temps-${RHT_OCP4_DEV_USER}-ocp.${RHT_OCP4_WILDCARD_DOMAIN}  ...
```

- 3.2. Verifizieren Sie, dass die Anwendung zur Temperaturumrechnung funktioniert, indem Sie einen Webbrowser öffnen und zu der im vorherigen Schritt angezeigten URL navigieren.



Bewertung

Führen Sie auf `workstation` den Befehl `lab openshift-review grade` aus, um Ihre Arbeit zu bewerten. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf `workstation` den Befehl `lab openshift-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- OpenShift Container Platform speichert Definitionen jeder OpenShift- oder Kubernetes-Ressourceninstanz als Objekt im verteilten Datenbankservice etcd des Clusters. Zu den gängigen Ressourcentypen gehören: Pod, Persistentes Volume (PV), Anforderung für persistentes Volume (PVC), Service (SVC), Route, Bereitstellung, DeploymentConfig und Build-Konfiguration (BC).
- Verwenden Sie den OpenShift-Befehlszeilen-Client oc zum Erstellen, Ändern und Löschen von Projekten.
- Erstellen von Anwendungsressourcen innerhalb eines Projekts.
- Löschen, Prüfen, Bearbeiten und Exportieren von Ressourcen innerhalb eines Projekts.
- Prüfen von Protokollen aus Anwendungs-Pods, Bereitstellungen und Build-Vorgängen.
- Mit dem Befehl oc new-app können Anwendungs-Pods auf viele verschiedene Weisen erstellt werden: anhand des Source-to-Image-Prozesses (S2I) aus einem in einer Image-Registry gehosteten, bereits bestehenden Container-Image, aus Dockerfiles oder aus Quellcode.
- Source-to-Image (S2I) ist ein Tool, das die Erstellung eines Container-Images über Anwendungsquellcode erleichtert. Dieses Tool ruft Quellcode von einem Git-Repository ab, fügt den Quellcode auf Basis einer gewünschten Sprache oder Technologie in ein ausgewähltes Container-Image ein und erstellt ein neues Container-Image, das die zusammengestellte Anwendung ausführt.
- Eine Route stellt eine Verbindung zwischen einer öffentlichen IP-Adresse und einem DNS-Hostnamen zu einer internen Service-IP-Adresse her. Während Services den Netzwerkzugriff zwischen Pods innerhalb einer OpenShift-Instanz erlauben, ermöglichen Routen Benutzern und Anwendungen den Netzwerkzugriff auf Pods außerhalb der OpenShift-Instanz.
- Sie können Anwendungen über die OpenShift-Webkonsole erstellen, bereitstellen und überwachen.

Kapitel 7

Bereitstellen von Anwendungen mit mehreren Containern

Ziel

Bereitstellen von in Containern aufgeteilten Anwendungen mit mehreren Container-Images

Ziele

- Beschreibung der Überlegungen zur Containerisierung von Anwendungen mit mehreren Container-Images
- Bereitstellen einer Anwendung mit mehreren Containern auf OpenShift Platform
- Bereitstellen einer Anwendung in OpenShift mithilfe einer Vorlage

Abschnitte

- Überlegungen zu Anwendungen mit mehreren Containern (und angeleitete Übung)
- Bereitstellen einer Anwendung mit mehreren Containern in OpenShift (und angeleitete Übung)
- Bereitstellen einer Anwendung mit mehreren Containern in OpenShift mithilfe einer Vorlage (und angeleitete Übung)

Praktische Übung

- Bereitstellen von Anwendungen mit mehreren Containern

Überlegungen zu Anwendungen mit mehreren Containern

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Beschreibung der Überlegungen zur Containerisierung von Anwendungen mit mehreren Container-Images
- Nutzung der Networking-Konzepte in Containern
- Erstellen einer Anwendung mit mehreren Containern mit Podman
- Beschreibung der Architektur der To Do List-Anwendung

Nutzen von Anwendungen mit mehreren Containern

Die bisher gezeigten Beispiele in diesem Kurs haben bei einem einzelnen Container gut funktioniert. Eine komplexere Anwendung kann jedoch die Vorteile der Bereitstellung verschiedener Komponenten in verschiedenen Containern nutzen. Ein Beispiel ist hier eine Anwendung, die aus einer Front-End-Webanwendung, einem REST-Back-End und einem Datenbankserver besteht. Diese Komponenten können unterschiedliche Abhängigkeiten, Anforderungen und Lebenszyklen haben.

Obwohl die Container von Anwendungen mit mehreren Containern zwar manuell orchestriert werden können, bieten Kubernetes und OpenShift Tools zur Vereinfachung der Orchestrierung. Der Versuch, Dutzende oder Hunderte von Containern manuell zu verwalten, wird schnell kompliziert. In diesem Abschnitt wird wieder Podman verwendet, um eine einfache Anwendung mit mehreren Containern zu erstellen, um die zugrunde liegenden manuellen Schritte für die Container-Orchestrierung zu demonstrieren. In späteren Abschnitten verwenden Sie Kubernetes und OpenShift, um dieselben Anwendungs-Container zu orchestrieren.

Erkundung der Services in einer Anwendung mit mehreren Containern

Rootfull-Container

Podman verwendet Container Network Interface (CNI), um zwischen allen Containern im Host ein Software-defined Network (SDN) zu erstellen. Wenn nicht anders angegeben, weist CNI einem Container beim Start eine neue IP-Adresse zu.

Jeder Container stellt alle Ports für die anderen Container im selben SDN bereit. Auf diese Weise kann jederzeit innerhalb desselben Netzwerks auf Services zugegriffen werden. Die Container stellen die Ports nur durch eine explizite Konfiguration für andere Netzwerke bereit.

Aufgrund der dynamischen Natur von Container-IP-Adressen können Anwendungen für die Kommunikation mit Middleware-Services und anderen Anwendungsdiensten keine festen IP-Adressen bzw. feste DNS-Hostnamen verwenden. Container mit dynamischen IP-Adressen können ein Problem darstellen, wenn Sie Anwendungen mit mehreren Containern verwenden, da jeder Container mit den anderen kommunizieren muss, um die Services zu verwenden, von denen er abhängt.

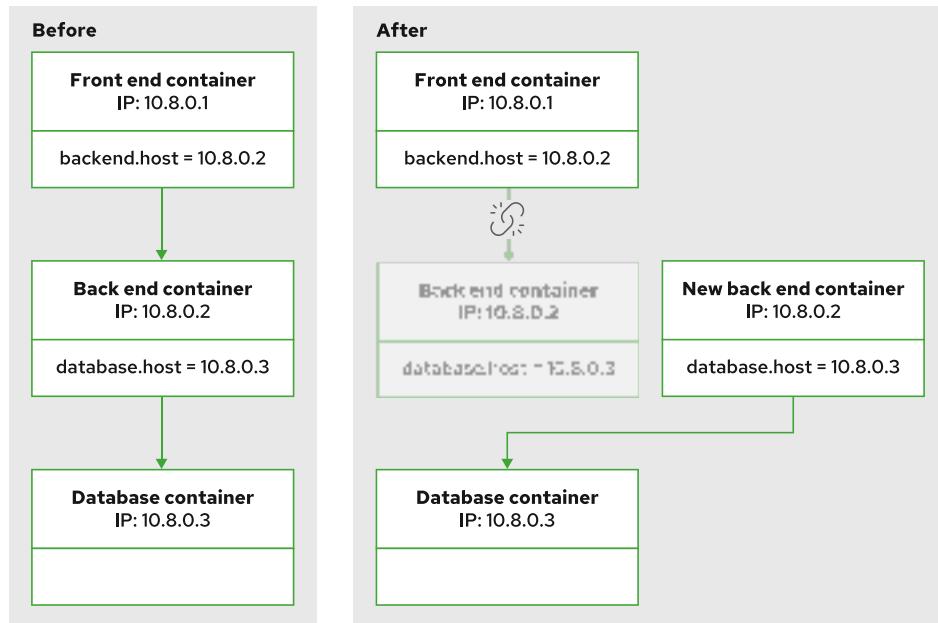


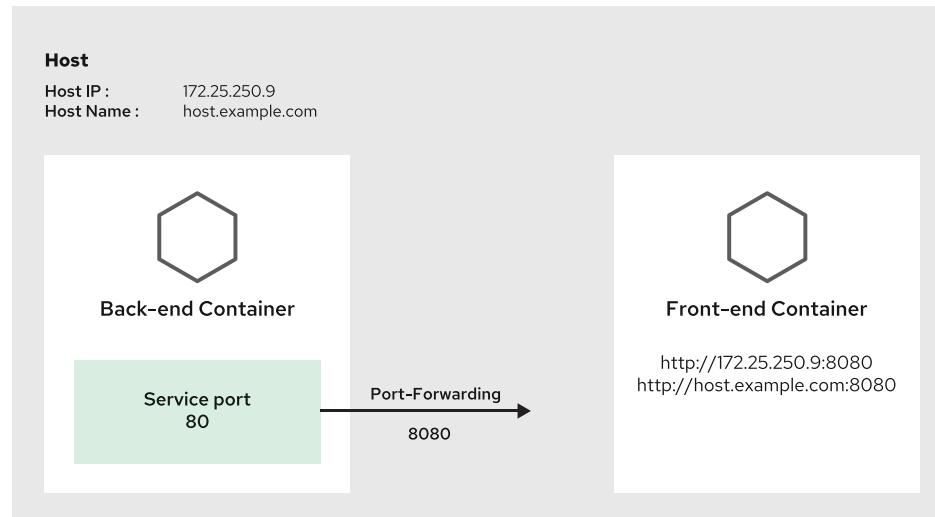
Abbildung 7.1: Bei einem Neustart werden Links für dreistufige Anwendungen unterbrochen

Denken Sie beispielsweise an eine Anwendung, die aus einem Front-End-Container, einem Back-End-Container und einer Datenbank besteht. Der Front-End-Container muss die IP-Adresse des Back-End-Containers abrufen. Gleiches gilt für den Back-End-Container bezüglich der IP-Adresse des Datenbank-Containers. Zudem kann sich die IP-Adresse ändern, wenn ein Container neu gestartet wird. Daher wird ein Prozess benötigt, der sicherstellt, dass jede Änderung der IP eine Aktualisierung der vorhandenen Container auslöst.

Kubernetes und OpenShift bieten potenzielle Lösungen für das Problem der Service-Ermittlung der dynamischen Art von Container-Netzwerken. Einige dieser Lösungen werden später in diesem Kapitel behandelt.

Rootless-Container

Rootless-Container unterstützen kein Software-Defined Network (SDN). Aus diesem Grund ist die IP-Adresse des Containers nicht für die Kommunikation mit anderen Containern auf dem Host verfügbar. Das Networking zwischen rootless-Containern lässt sich über die Portweiterleitung erreichen. Die Portweiterleitung ermöglicht den externen Zugriff auf einen Container-Service vom Host aus. Die Portweiterleitung wurde unter *Kapitel 3, Verwalten von Containern* erläutert.

**Abbildung 7.2: Anwendungen mit mehreren Containern für rootless-Container**

Denken Sie beispielsweise an eine Anwendung, die aus einem Front-End-Container und einem Back-End-Container besteht. Der Back-End-Container verfügt über einen Service, der auf Port 80 ausgeführt wird. Auf diesen Service/Port kann von außerhalb des Containers nicht zugegriffen werden. Der Front-End-Container muss vom Back-End-Container aus auf den Service/Port zugreifen. Der erste Schritt besteht in der Verwendung der *Portweiterleitung* für den erforderlichen Service. In diesem Beispiel haben wir Service-Port 80 an 8080 weitergeleitet. Nun kann der Front-End-Container über die Host-IP-Adresse und den weitergeleiteten Port auf den Service/Port des Back-End-Containers zugreifen.

Beschreiben der Anwendung „To Do List“

In vielen Labs aus diesem Kurs wird eine To Do List-Anwendung eingesetzt. Diese Anwendung ist in drei Stufen eingeteilt, wie in der folgenden Abbildung dargestellt:

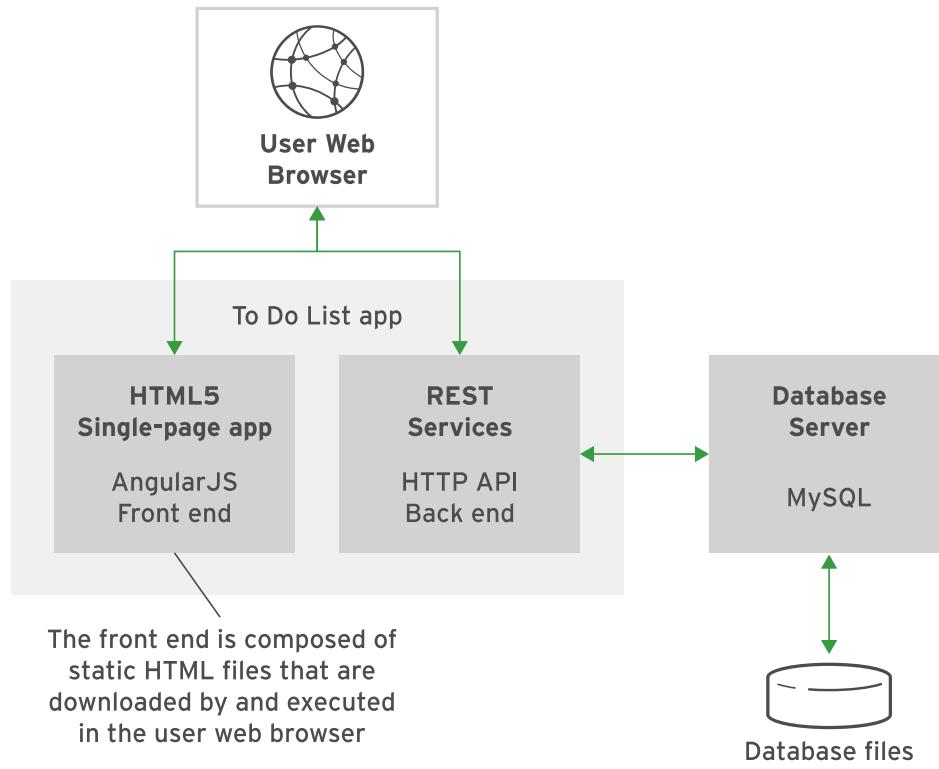


Abbildung 7.3: Logische Architektur der To Do List-Anwendung

- Die Präsentationsstufe wird als eine einzelne HTML5-Front-End-Seite mithilfe von AngularJS erstellt.
- Die Unternehmensstufe wird als HTTP-API-Back-End mithilfe von Node.js erstellt.
- Die Persistenzstufe basiert auf einem MySQL-Datenbankserver.

Die folgende Abbildung zeigt einen Bildschirmabschnitt der Anwendungsoberfläche:

To Do List Application

To Do List

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

FirstPrevious1NextLast

Add Task

Description:

Add Description.

Completed:

ClearSave

Abbildung 7.4: Die To Do List-Anwendung

Links in der Tabelle stehen die Elemente, die erledigt werden müssen. Auf der rechten Seite wird ein Formular angezeigt, mit dem Sie ein neues Element hinzufügen können.

Der Private Kursraum-Registry-Server services.lab.example.com stellt Anwendungen in zwei Versionen bereit:

nodejs

Stellt dar, wie ein typischer Entwickler die Anwendung als einzelne Einheit erstellen würde, ohne diese in Stufen oder Services aufzuteilen.

nodejs_api

Zeigt die Änderungen, die erforderlich sind, um die Anwendung in Präsentations- und Unternehmensstufen aufzuteilen. Jede Stufe entspricht einem isolierten Container-Image.

Die Quellen beider Anwendungsversionen sind im Ordner todoapp/nodejs im Git-Repository unter <https://github.com/RedHatTraining/D0180-apps.git> verfügbar.

► Angeleitete Übung

Bereitstellen von Webanwendung und MySQL in Linux-Containern

In diesem Lab erstellen Sie ein Skript, das ausgeführt wird und den Anwendungs-Container „Node.js“ mit dem MySQL-Container vernetzt.

Ergebnisse

Sie sollten in der Lage sein, Container für die Erstellung einer Anwendung mit mehreren Ebenen zu vernetzen.

Bevor Sie Beginnen

Sie müssen den Quellcode der Anwendung To Do List und die Lab-Dateien auf workstation besitzen. Führen Sie den folgenden Befehl aus, um die Umgebung für diese Übung einzurichten:

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

Anweisungen

- 1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an. Lesen Sie die Anweisungen in *Anhang D, Erstellen eines Red Hat-Benutzerkontos*, wenn Sie sich bei Red Hat registrieren müssen.

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

- 2. Überprüfen Sie das Containerfile.
Öffnen Sie mit Ihrem bevorzugten Editor das fertige Containerfile unter /home/student/D0180/labs/multicontainer-design/deploy/nodejs/Containerfile, und überprüfen Sie es.
- 3. Führen Sie den Befehl `ip addr` aus, um grep für die Host-IP-Adresse auszuführen.

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

- 4. Untersuchen Sie die Umgebungsvariablen.
Überprüfen Sie die Umgebungsvariablen, die dem REST-API-Container „Node.js“ die Kommunikation mit dem MySQL-Container ermöglichen.

- 4.1. Zeigen Sie die Datei /home/student/D0180/labs/multicontainer-design/deploy/nodejs/nodejs-source/models/db.js an, die die unten angegebene Datenbankkonfiguration enthält:

```
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. Beachten Sie die von der REST-API verwendeten Umgebungsvariablen. Diese Variablen werden unter Verwendung der -e-Optionen mit dem Befehl podman run in dieser angeleiteten Übung bereitgestellt. Diese Umgebungsvariablen werden weiter unten beschrieben.

MYSQL_DATABASE

Der Name der MySQL-Datenbank im mysql-Container.

MYSQL_USER

Der Name des Datenbankbenutzers, den der todoapi-Container verwendet, um MySQL-Befehle auszuführen.

MYSQL_PASSWORD

Das Passwort des Datenbankbenutzers, den der todoapi-Container für die Authentifizierung beim mysql-Container verwendet.



Anmerkung

Die Host- und Port-Details des MySQL-Containers sind in die REST-API-Anwendung eingebettet. Der Host, wie oben in der Datei db.js gezeigt, ist die IP-Adresse des Hosts, für die wir im vorherigen Schritt grep ausgeführt haben.

- 5. Erstellen Sie mithilfe des bereitgestellten Containerfiles das untergeordnete Image der Anwendung To Do List.

- 5.1. Erstellen Sie das untergeordnete Image.

Untersuchen Sie das Skript /home/student/D0180/labs/multicontainer-design/deploy/nodejs/build.sh, um zu erfahren, wie das Image erstellt wird. Führen Sie die folgenden Befehle aus, um das untergeordnete Image zu erstellen.

```
[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

```



Anmerkung

Das Skript `build.sh` lockert die Einschränkungen für den Schreibzugriff auf das Build-Verzeichnis, um die Installation von Abhängigkeiten durch Nicht-Root-Benutzer zu ermöglichen.

Manchmal führt eine Netzwerkinstabilität zu einem `ERR!` für das Build-Skript. Um dieses Problem zu beheben, führen Sie das Build-Skript erneut aus.

- 5.2. Warten Sie, bis der Erstellungsvorgang abgeschlossen ist. Führen Sie anschließend den folgenden Befehl aus, um zu verifizieren, dass das Image erstellt wurde:

```
[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. Ändern Sie das vorhandene Skript, um Container mit der entsprechenden Ports zu erstellen, wie im vorherigen Schritt definiert. Im Skript sind die Befehle so angeordnet, dass zuerst der mysql-Container und dann der todoapi-Container gestartet werden, bevor eine Verbindung zum mysql-Container hergestellt wird. Nach dem Aufruf jedes Containers folgt eine Wartezeit von 9 Sekunden, sodass jeder Container über genügend Zeit für den Start verfügt.

- 6.1. Bearbeiten Sie die `run.sh`-Datei unter `/home/student/D0180/labs/multicontainer-design/deploy/nodejs/networked`, um den Befehl `podman run` in die entsprechende Zeile für das Aufrufen des mysql-Containers einzufügen. Auf dem folgenden Bildschirm wird der genaue Podman-Befehl angezeigt, der in die Datei eingefügt werden muss.

```
podman run -d --name mysql -e MYSQL_DATABASE=items -e MYSQL_USER=user1 \
-e MYSQL_PASSWORD=mypassword -e MYSQL_ROOT_PASSWORD=r00tpass \
-v $PWD/work/data:/var/lib/mysql/data \
-p 3306:3306 \
registry.redhat.io/rhel8/mysql-8.0:1
```

Im vorherigen Befehl wurden `MYSQL_DATABASE`, `MYSQL_USER` und `MYSQL_PASSWORD` mit den Anmeldeinformationen für den Zugriff auf die MySQL-Datenbank gefüllt. Diese Umgebungsvariablen werden benötigt, damit der mysql-Container ausgeführt werden kann. Darüber hinaus wird der lokale Ordner `$PWD/work/data` als Volume im Dateisystem des Containers eingebunden.

- 6.2. Fügen Sie in die gleiche `run.sh`-Datei in der entsprechenden Zeile einen weiteren `podman run`-Befehl ein, um den `todoapi`-Container auszuführen. Auf dem folgenden Bildschirm wird der Befehl `docker` angezeigt, der in die Datei eingefügt werden muss.

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



Anmerkung

Stellen Sie sicher, dass nach jedem Befehl `podman run`, der in das `run.sh`-Skript eingefügt wurde, der Befehl `sleep 9` folgt. Wenn Sie diesen Schritt wiederholen müssen, müssen das `work`-Verzeichnis und seine Inhalte vor der Ausführung des `run.sh`-Skripts gelöscht werden.

- 6.3. Überprüfen Sie, ob das `run.sh`-Skript mit dem Lösungsskript unter `/home/student/D0180/solutions/multicontainer-design/deploy/nodejs/networked/run.sh` übereinstimmt.
- 6.4. Speichern Sie die Datei und beenden Sie den Editor.

► 7. Führen Sie die Container aus.

- 7.1. Führen Sie den folgenden Befehl aus, um das Skript auszuführen, das Sie aktualisiert haben, um die Container `mysql` und `todoapi` auszuführen.

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

- 7.2. Überprüfen Sie, ob die Container gestartet wurden.

```
[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. Füllen Sie die Datenbank `items` anhand der Tabelle `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. Überprüfen Sie die Umgebungsvariablen des API-Containers.

Führen Sie den folgenden Befehl aus, um die Umgebungsvariablen im API-Container zu untersuchen.

```
[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. Testen Sie die Anwendung

- 10.1. Führen Sie den Befehl `curl` aus, um die REST-API für die Anwendung To Do List zu überprüfen.

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

Wenn Sie die Option `-w "\n"` mit dem Befehl `curl` verwenden, wird die Shell-Eingabeaufforderung in der nächsten Zeile angezeigt und nicht mit der Ausgabe in derselben Zeile vermischt.

- 10.2. Öffnen Sie Firefox auf `workstation`, und gehen Sie zu `http://127.0.0.1:30080/todo/`. Die Aufgabenliste wird angezeigt.



Anmerkung

Hängen Sie einen Schrägstrich (/) an.

- 10.3. Wechseln Sie zum Verzeichnis „/home/student/“.

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

Beenden

Führen Sie auf `workstation` das Skript `lab multicontainer-design finish` aus, um diese Übung zu beenden.

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

Hiermit ist die angeleitete Übung beendet.

Bereitstellen einer Anwendung mit mehreren Containern auf OpenShift

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Beschreiben der Unterschiede zwischen Podman und Kubernetes
- Bereitstellen einer Anwendung mit mehreren Containern in OpenShift

Vergleich Podman und Kubernetes

Wenn Sie Umgebungsvariablen verwenden, können Sie Informationen zwischen Containern mit Podman teilen. Es gibt dennoch einige Einschränkungen, und um sicherzustellen, dass alle Umgebungsvariablen synchronisiert werden – insbesondere dann, wenn Sie mit vielen Containern arbeiten – müssen viele Aufgaben manuell durchgeführt werden. Kubernetes bietet einen Ansatz zur Lösung dieses Problems, wenn Sie Services für Ihre Container erstellen, wie in vorherigen Kapiteln erörtert.

Services in Kubernetes

Pods werden an den Kubernetes-Namespace angehängt; dies wird in OpenShift als *Projekt* bezeichnet. Beim Starten eines Pods fügt Kubernetes automatisch eine Reihe von Umgebungsvariablen für jeden Service hinzu, der im selben Namespace definiert wurde.

Jeder Service, der in Kubernetes definiert ist, erzeugt Umgebungsvariablen für die IP-Adresse und die Port-Nummer, unter der der Service verfügbar ist. Kubernetes fügt den Containern diese Umgebungsvariablen automatisch über Pods, die sich im selben Namespace befinden, hinzu. Diese Umgebungsvariablen folgen in der Regel einer Konvention:

- **Großschreibung:** Alle Umgebungsvariablen werden in Großbuchstaben angegeben.
- **Snakecase:** Jede von einem Service erstellte Umgebungsvariable setzt sich normalerweise aus mehreren Wörtern zusammen, die durch einen Unterstrich (_) getrennt sind.
- **Servicename an erster Stelle:** Das erste Wort, das von einem Service für eine Umgebungsvariable erstellt wird, ist der Servicename.
- **Protokolltyp:** Die meisten Netzwerkumgebungsvatiablen enthalten den Protokolltyp (TCP oder UDP).

Dies sind die Umgebungsvariablen, die von Kubernetes für einen Service erzeugt werden:

- <SERVICE_NAME>_SERVICE_HOST: Repräsentiert die IP-Adresse, die von einem Service für den Zugriff auf einen Pod aktiviert wird.
- <SERVICE_NAME>_SERVICE_PORT: Repräsentiert den Port, an dem der Serverport aufgeführt wird.
- <SERVICE_NAME>_PORT: Repräsentiert die Adresse, den Port und das Protokoll, die von dem Service für den externen Zugriff bereitgestellt werden.
- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>: Definiert einen Alias für den <SERVICE_NAME>_PORT.

- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PROTO: Identifiziert den Protokolltyp (TCP oder UDP).
- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_PORT: Definiert einen Alias für den <SERVICE_NAME>_SERVICE_PORT.
- <SERVICE_NAME>_PORT_<PORT_NUMBER>_<PROTOCOL>_ADDR: Definiert einen Alias für den <SERVICE_NAME>_SERVICE_HOST.

Dies sind die Umgebungsvariablen, die von Kubernetes für einen Service erzeugt werden.

Angenommen, es liegt der folgende Service vor:

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mysql
  name: mysql
spec:
  ports:
    - protocol: TCP
      port: 3306
  selector:
    name: mysql
```

Die folgenden Umgebungsvariablen sind für jeden Pod nach der Erstellung des Services auf demselben Namespace verfügbar:

```
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
```



Anmerkung

Andere relevante <SERVICE_NAME>_PORT_*-Umgebungsvariablennamen werden anhand des Protokolls festgelegt. IP-Adresse und Portnummer werden in der <SERVICE_NAME>_PORT-Umgebungsvariablen festgelegt. Zum Beispiel: Aus dem Eintrag MYSQL_PORT=tcp://10.0.0.11:3306 wird die Erstellung von Umgebungsvariablen mit Namen wie MYSQL_PORT_3306_TCP, MYSQL_PORT_3306_TCP_PROTO, MYSQL_PORT_3306_TCP_PORT und MYSQL_PORT_3306_TCP_ADDR abgeleitet. Wenn die Protokollkomponente einer Umgebungsvariablen nicht definiert ist, verwendet Kubernetes das TCP-Protokoll und weist die Variablennamen entsprechend zu.

► Angeleitete Übung

Erstellen einer Anwendung in OpenShift

In dieser Übung stellen Sie die Anwendung To Do List in OpenShift Container Platform bereit.

Ergebnisse

Sie sollten in der Lage sein, in OpenShift Container Platform eine Anwendung zu erstellen und bereitzustellen.

Bevor Sie Beginnen

Sie müssen den Quellcode der Anwendung To Do List und die Lab-Dateien auf workstation besitzen. Führen Sie den folgenden Befehl in einem neuen Terminalfenster aus, um die Lab-Dateien herunterzuladen, und um den Status des OpenShift-Clusters zu verifizieren:

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

Anweisungen

- 1. Erstellen Sie die Anwendung To Do List aus der bereitgestellten YAML-Datei.

- 1.1. Melden Sie sich bei OpenShift Container Platform an.

```
[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".
```

Wenn der Befehl `oc login` Sie bezüglich unsicherer Verbindungen zu einer Eingabe auffordert, antworten Sie mit „y“ (ja).

- 1.2. Erstellen Sie ein neues Projekt *application* in OpenShift, um es in dieser Übung zu verwenden. Führen Sie folgenden Befehl aus, um das Projekt *application* zu erstellen.

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

- 1.3. Überprüfen Sie die YAML-Datei.

Öffnen Sie mit Ihrem bevorzugten Editor die App-Datei unter `/home/student/D0180/labs/multicontainer-application/todo-app.yml`, und überprüfen

Kapitel 7 | Bereitstellen von Anwendungen mit mehreren Containern

Sie sie. Beachten Sie die folgenden Ressourcen, die in `todo-app.yml` definiert sind, und überprüfen Sie die entsprechenden Konfigurationen.

- Die `todoapi`-Pod-Definition definiert die Node.js-Anwendung.
- Die `MySQL`-Pod-Definition definiert die MySQL-Datenbank.
- Der `todoapi`-Service stellt eine Verbindung zum Node.js-Anwendungs-Pod bereit.
- Der `mysql`-Service bietet Konnektivität zum MySQL-Datenbank-Pod.
- Die `dbclaim`-Definition für die Anforderung des persistenten Volumens definiert das MySQL-Volume `/var/lib/mysql/data`.

1.4. Erstellen Sie Anwendungsressourcen mit der angegebenen yaml-Datei.

Erstellen Sie die Anwendungsressourcen mit dem Befehl `oc create`. Führen Sie im Terminalfenster den folgenden Befehl aus:

```
[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. Überprüfen Sie die Bereitstellung.

Überprüfen Sie den Status der Bereitstellung mithilfe des Befehls `oc get pods` und der Option `-w`, um den Pod-Status weiterhin zu überwachen. Warten Sie, bis beide Container ausgeführt werden. Das Starten beider Pods kann einige Zeit in Anspruch nehmen.

```
[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
```

Drücken Sie `Strg+C`, um den Befehl zu verlassen.

► **2.** Stellen Sie eine Verbindung zum MySQL-Datenbankserver her, und füllen Sie die Item-Datenbank mit den Daten.

2.1. Konfigurieren Sie auf dem Rechner `workstation` die Portweiterleitung zwischen `workstation` und dem Datenbank-Pod, der unter OpenShift mit Port 3306 ausgeführt wird. Das Terminal bleibt hängen, nachdem der Befehl ausgeführt wurde.

```
[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. Öffnen Sie auf dem Rechner „`workstation`“ ein weiteres Terminal, und füllen Sie den MySQL-Server mithilfe des MySQL-Clients mit den Daten.

```
[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. Schließen Sie das Terminal, und kehren Sie zum vorherigen zurück. Beenden Sie die Portweiterleitung. Drücken Sie dazu Strg+C.

```
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
Handling connection for 3306
^C
```

► 3. Stellen Sie den Service bereit.

Damit die Anwendung To Do Liste über den OpenShift-Router zugänglich und als öffentlicher FQDN verfügbar ist, verwenden Sie den Befehl oc expose, um den todoapi-Dienst verfügbar zu machen.

Führen Sie im Terminalfenster den folgenden Befehl aus.

```
[student@workstation multicontainer-application]$ oc expose service todoapi
route.route.openshift.io/todoapi exposed
```

► 4. Testen Sie die Anwendung

- 4.1. Suchen Sie den FQDN der Anwendung, indem Sie den Befehl oc status ausführen und den FQDN für die App notieren.

Führen Sie im Terminalfenster den folgenden Befehl aus.

```
[student@workstation multicontainer-application]$ oc status | grep -o "http:.com"
http://todoapi-$RHT_OCP4_QUAY_USER-application.${RHT_OCP4_WILDCARD_DOMAIN}
```

- 4.2. Verwenden Sie curl, um die REST-API für die Anwendung To Do List zu testen.

```
[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}
```

Wenn Sie die Option -w "\n" mit dem Befehl curl verwenden, wird die Shell-Eingabeaufforderung in der nächsten Zeile angezeigt und nicht mit der Ausgabe in derselben Zeile vermischt.

- 4.3. Wechseln Sie zum Verzeichnis „/home/student/“.

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

- 4.4. Öffnen Sie Firefox auf workstation, und navigieren Sie mit Ihrem Browser zu http://todoapi-\$RHT_OCP4_QUAY_USER-application.\${RHT_OCP4_WILDCARD_DOMAIN}/todo/. Die Anwendung To Do List sollte angezeigt werden.



Anmerkung

Der abschließende Schrägstrich in der oben erwähnten URL ist notwendig. Wenn Sie dies in der URL nicht angeben, können Probleme mit der Anwendung auftreten.

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

First Previous 1 Next Last

Add Task

Description: Add Description.

Completed:

Clear Save

Abbildung 7.5: Anwendung „To Do List“

Beenden

Führen Sie auf workstation das Skript `lab multicontainer-application finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit ist die angeleitete Übung beendet.

Bereitstellen einer Anwendung mit mehreren Containern in OpenShift mithilfe einer Vorlage

Am Ende dieses Abschnitts sollten Teilnehmer in der Lage sein, eine Anwendung mit mehreren Containern unter Verwendung einer Vorlage auf OpenShift bereitzustellen.

Überprüfen des Gerüsts einer Vorlage

Für die Bereitstellung einer Anwendung auf OpenShift Container Platform müssen häufig mehrere zugehörige Ressourcen in einem Projekt erstellt werden. Beispielsweise sind für die Ausführung einer Webanwendung in einem OpenShift-Projekt möglicherweise die Ressourcen `BuildConfig`, `Deployment`, `Service` und `Route` erforderlich. Oft haben die Attribute dieser Ressourcen denselben Wert, beispielsweise das Namensattribut einer Ressource.

OpenShift-Vorlagen bieten eine Möglichkeit, die Erstellung von Ressourcen zu vereinfachen, die eine Anwendung benötigt. Eine Vorlage definiert eine Reihe zusammengehöriger Ressourcen, die gemeinsam erstellt werden sollen, sowie eine Reihe von Anwendungsparametern. Die Attribute von Vorlagenressourcen werden normalerweise in Form von Vorlagenparametern definiert, beispielsweise dem Namensattribut einer Ressource.

Eine Anwendung könnte beispielsweise aus einer Front-End-Webanwendung und einem Datenbankserver bestehen. Dabei umfasst jede eine Serviceressource und eine Bereitstellungsressource. Sie teilen eine Reihe von Anmeldeinformationen (Parameter) für das Front-End zur Authentifizierung beim Back-End. Vorlagen können durch die Angabe von Parametern verarbeitet werden. Alternativ können Sie automatisch erstellt werden (z. B. für ein eindeutiges Datenbankpasswort), um die Ressourcenliste in der Vorlage als in sich geschlossene Anwendung zu instanzieren.

Das OpenShift-Installationsprogramm erstellt standardmäßig mehrere Vorlagen im `OpenShift`-Namespace. Führen Sie den Befehl `oc get templates` mit der Option `-n openshift` aus, um diese vorinstallierten Vorlagen aufzulisten:

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

Die YAML-Definition einer Vorlage kann entsprechend den Anforderungen Ihrer Anwendungen geändert werden. Dies führen Sie in einer folgenden Übung durch.

Im Folgenden wird eine YAML-Vorlagendefinition gezeigt:

```
[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 Definiert eine Liste beliebiger Tags, die dieser Vorlage zugeordnet werden sollen. Geben Sie eines dieser Tags auf die Benutzeroberfläche ein, um diese Vorlage zu finden.
- ➋ Definiert den Vorlagennamen.
- ➌ Der Abschnitt `objects` definiert die Liste der OpenShift-Ressourcen für diese Vorlage. Diese Vorlage erstellt vier Ressourcen: `Secret`, `Service`, `PersistentVolumeClaim` und `Deployment`.
- ➍ Der Name aller vier Ressourcenobjekte ist auf den Wert des Parameters `DATABASE_SERVICE_NAME` festgelegt.
- ➎ Der Abschnitt `parameters` enthält eine Liste von neun Parametern.
- ➏ Vorlagenressourcen definieren ihre Attribute häufig anhand der Werte dieser Parameter, wie dies mit dem Parameter `DATABASE_SERVICE_NAME` gezeigt wird.
- ➐ Sofern Sie keinen Wert für den Parameter `MYSQL_PASSWORD` angeben, wenn Sie eine Anwendung mit dieser Vorlage erstellen, generiert OpenShift ein Passwort, das diesem regulären Ausdruck entspricht.

Sie können eine neue Vorlage im OpenShift-Cluster veröffentlichen, sodass andere Entwickler anhand der Vorlage eine Anwendung erstellen können.

Angenommen, Sie verfügen über eine Aufgabenlistenanwendung mit dem Namen `todo`, die ein Deployment-, Service- und Route-Objekt von OpenShift für die Bereitstellung benötigt. Sie erstellen eine YAML-Vorlagendefinitionsdatei, die Attribute für diese OpenShift-Ressourcen sowie Definitionen für alle erforderlichen Parameter definiert. Führen Sie unter der Annahme, dass die Vorlage in der Datei `todo-template.yaml` definiert ist, den Befehl `oc create` aus, um die Anwendungsvorlage zu veröffentlichen:

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

Die Vorlage wird standardmäßig für das aktuelle Projekt erstellt, es sei denn, Sie geben mithilfe der Option `-n` ein anderes Projekt an (siehe folgendes Beispiel):

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



Wichtig

Jede Vorlage, die unter dem OpenShift-Namespace (OpenShift-Projekt) erstellt wurde, ist in der Web Console unter dem Dialogfeld verfügbar, auf das über **CatalogDeveloper** → **Catalog** zugegriffen werden kann. Darüber hinaus kann auf jede Vorlage, die unter dem aktuellen Projekt erstellt wird, von diesem Projekt aus zugegriffen werden.

Parameter

Vorlagen definieren eine Reihe von Parametern, denen Werte zugewiesen sind. OpenShift-Ressourcen, die in der Vorlage definiert sind, können die zugehörigen Konfigurationswerte durch Verweis auf die *benannten Parameter* abrufen. Die Parameter in einer Vorlage können Standardwerte haben, diese sind aber optional. Standardwerte können beim Bearbeiten der Vorlage ersetzt werden.

Jeder Parameterwert kann entweder explizit mithilfe des Befehls `oc process` festgelegt oder durch OpenShift der Parameterkonfiguration entsprechend erstellt werden.

Zum Auflisten der verfügbaren Parameter aus einer Vorlage gibt es zwei Möglichkeiten. Die erste verwendet den Befehl `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 (5.7, or latest).
  Required:  true
  Value:    5.7

Object Labels:  template=mysql-persistent-template

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

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

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

```
Objects:
Secret      ${DATABASE_SERVICE_NAME}
Service     ${DATABASE_SERVICE_NAME}
PersistentVolumeClaim ${DATABASE_SERVICE_NAME}
Deployment   ${DATABASE_SERVICE_NAME}
```

Bei der zweiten Methode wird `oc process` mit der Option `--parameters` verwendet:

NAME	DESCRIPTION	GENERATOR	VALUE
MEMORY_LIMIT	Maximum amount of memory to allocate to the application		512Mi
NAMESPACE	The OpenShift namespace where the application will run		openshift
DATABASE_SERVICE_NAME	The name of the MySQL service to connect to		mysql
MYSQL_USER	Username for the MySQL connection	expression	user[A-Z0-9]{3}
MYSQL_PASSWORD	Password for the MySQL connection	expression	[a-zA-Z0-9]{16}
MYSQL_ROOT_PASSWORD	Password for the MySQL root user	expression	[a-zA-Z0-9]{16}
MYSQL_DATABASE	Name of the MySQL database to use		sampled
VOLUME_CAPACITY	Volume size for the MySQL persistent volume		1Gi
MYSQL_VERSION	Version of MySQL to use		5.7

Bearbeiten einer Vorlage über die Befehlszeilenschnittstelle

Beim Verarbeiten einer Vorlage generieren Sie eine Ressourcenliste, um eine neue Anwendung zu erstellen. Führen Sie den Befehl `oc process` aus, um eine Vorlage zu verarbeiten:

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

Mit dem vorherigen Befehl wird eine Vorlagendatei im JSON- oder YAML-Format verarbeitet und die Liste der Ressourcen als Standardausgabe zurückgegeben. Das Format der Ausgaberessourcenliste ist JSON. Verwenden Sie zur Ausgabe der Ressourcenliste im YAML-Format `-o yaml` mit dem Befehl `oc process`:

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

Alternativ können Vorlagen über das aktuelle Projekt oder das `openshift`-Projekt verarbeitet werden:

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



Anmerkung

Der Befehl `oc process` gibt die Ressourcenliste als Standardausgabe zurück. Diese Ausgabe kann an eine Datei umgeleitet werden:

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

Kapitel 7 | Bereitstellen von Anwendungen mit mehreren Containern

Vorlagen generieren häufig Ressourcen mit konfigurierbaren Attributen, die auf den Vorlagenparametern basieren. Um einen Parameter zu überschreiben, verwenden Sie die Option -p, gefolgt von einem <name>=<value>-Paar.

```
[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
```

Erstellen Sie die Anwendung unter Verwendung der generierten YAML-Ressourcendefinitionsdatei:

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

Es besteht auch die Möglichkeit, die Vorlage mithilfe einer UNIX-Pipe zu verarbeiten und die Anwendung zu erstellen, ohne eine Ressourcendefinitionsdatei zu speichern:

```
[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 -
```

Zum Verwenden einer Vorlage im openshift-Projekt, um eine Anwendung in Ihrem Projekt zu erstellen, exportieren Sie zunächst die Vorlage:

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

Identifizieren Sie als Nächstes geeignete Werte für die Vorlagenparameter, und verarbeiten Sie die Vorlage:

```
[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 -
```

Sie können auch zwei Schrägstriche (>//) verwenden, um den Namespace als Teil des Vorlagennamens bereitzustellen:

```
[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 -
```

Alternativ können Sie eine Anwendung mithilfe des Befehls oc new-app erstellen, indem Sie den Vorlagennamen als das Optionsargument --template weiterleiten:

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



Literaturhinweise

Entwicklerinformationen zu Vorlagen finden Sie im Abschnitt *Using Templates* der OpenShift Container Platform-Dokumentation:

Entwicklerleitfaden

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/images/index#using-templates

► Angeleitete Übung

Erstellen einer Anwendung mit einer Vorlage

In dieser Übung stellen Sie die Anwendung To Do List in OpenShift Container Platform unter Verwendung einer Vorlage bereit, um die Ressourcen zu definieren, die Ihre Anwendung ausführen muss.

Ergebnisse

Sie sollten in der Lage sein, in OpenShift Container Platform eine Anwendung unter Verwendung einer bereitgestellten JSON-Vorlage zu erstellen und bereitzustellen.

Bevor Sie Beginnen

Sie müssen den Quellcode der Anwendung To Do List und die Lab-Dateien auf workstation besitzen. Führen Sie den folgenden Befehl in einem neuen Terminalfenster aus, um die Lab-Dateien herunterzuladen, und um den Status des OpenShift-Clusters zu verifizieren:

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

Anweisungen

- 1. Erstellen Sie die Aufgabenliste aus der bereitgestellten JSON-Vorlage.

- 1.1. Melden Sie sich bei OpenShift Container Platform an.

```
[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".
```

Wenn der Befehl `oc login` Sie bezüglich unsicherer Verbindungen zu einer Eingabe auffordert, antworten Sie mit „y“ (ja).

- 1.2. Erstellen Sie in OpenShift eine neue Projektvorlage für diese Übung. Führen Sie den folgenden Befehl aus, um das Vorlagenprojekt zu erstellen.

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

- 1.3. Überprüfen Sie die Vorlage.

Öffnen Sie mit Ihrem bevorzugten Editor die Vorlage unter `/home/student/DO180/labs/multicontainer-openshift/todo-template.json`, und

Kapitel 7 | Bereitstellen von Anwendungen mit mehreren Containern

überprüfen Sie sie. Beachten Sie die folgenden Ressourcen, die in der Vorlage definiert sind, und überprüfen Sie die entsprechenden Konfigurationen.

- Die todoapi-Pod-Definition definiert die Node.js-Anwendung.
 - Die MySQL-Pod-Definition definiert die MySQL-Datenbank.
 - Der todoapi-Service stellt eine Verbindung zum Node.js-Anwendungs-Pod bereit.
 - Der mysql-Service bietet Konnektivität zum MySQL-Datenbank-Pod.
 - Die dbclaim-Definition für die Anforderung des persistenten Volumens definiert das MySQL-Volume /var/lib/mysql/data.
14. Verarbeiten Sie die Vorlage und erstellen Sie die Anwendungsressourcen.

Führen Sie den Befehl `oc process` aus, um die Vorlagendatei zu verarbeiten. Für diese Vorlage muss der Quay.io-Namespace die Container-Images abrufen. Führen Sie den Befehl `pipe` aus, um das Ergebnis an den Befehl `oc create` zu senden.

Führen Sie im Terminalfenster den folgenden Befehl aus:

```
[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
```

15. Überprüfen Sie die Bereitstellung.

Überprüfen Sie den Status der Bereitstellung mithilfe des Befehls `oc get pods` und der Option `-w`, um den Pod-Status weiterhin zu überwachen. Warten Sie, bis beide Container ausgeführt werden. Das Starten beider Pods kann einige Zeit in Anspruch nehmen.

```
[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
```

Drücken Sie Strg+C, um den Befehl zu verlassen.

- 2. Stellen Sie eine Verbindung zum MySQL-Datenbankserver her, und füllen Sie die Item-Datenbank mit den Daten.
- 2.1. Konfigurieren Sie auf dem Rechner `workstation` die Portweiterleitung zwischen `workstation` und dem Datenbank-Pod, der unter OpenShift mit Port 3306 ausgeführt wird. Das Terminal bleibt hängen, nachdem der Befehl ausgeführt wurde.

```
[student@workstation multicontainer-openshift]$ oc port-forward mysql 3306:3306
Forwarding from 127.0.0.1:3306 -> 3306
Forwarding from [::1]:3306 -> 3306
```

Kapitel 7 | Bereitstellen von Anwendungen mit mehreren Containern

- 2.2. Öffnen Sie auf dem Rechner „workstation“ ein weiteres Terminal, und füllen Sie den MySQL-Server mithilfe des MySQL-Clients mit den Daten.

```
[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. Schließen Sie das Terminal, und kehren Sie zum vorherigen zurück. Beenden Sie die Portweiterleitung. Drücken Sie dazu Strg+C.

```
Forwarding from 127.0.0.1:3306 -> 3306  
Forwarding from [::1]:3306 -> 3306  
Handling connection for 3306  
^C
```

► 3. Stellen Sie den Service bereit.

Damit die Anwendung To Do Liste über den OpenShift-Router zugänglich und als öffentlicher FQDN verfügbar ist, verwenden Sie den Befehl `oc expose`, um den `todoapi`-Dienst verfügbar zu machen.

Führen Sie im Terminalfenster den folgenden Befehl aus.

```
[student@workstation multicontainer-openshift]$ oc expose service todoapi  
route.route.openshift.io/todoapi exposed
```

► 4. Testen Sie die Anwendung

- 4.1. Suchen Sie den FQDN der Anwendung, indem Sie den Befehl `oc status` ausführen und den FQDN für die App notieren.

Führen Sie im Terminalfenster den folgenden Befehl aus.

```
[student@workstation multicontainer-openshift]$ oc status | grep -o "http:.com"  
http://todoapi-${RHT_OCP4_DEV_USER}-template.${RHT_OCP4_WILDCARD_DOMAIN}
```

- 4.2. Verwenden Sie `curl`, um die REST-API für die Anwendung To Do List zu testen.

```
[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}
```

Wenn Sie die Option `-w "\n"` mit dem Befehl `curl` verwenden, wird die Shell-Eingabeaufforderung in der nächsten Zeile angezeigt und nicht mit der Ausgabe in derselben Zeile vermischt.

- 4.3. Wechseln Sie zum Verzeichnis „/home/student/“.

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

- 4.4. Öffnen Sie Firefox auf `workstation`, und navigieren Sie mit Ihrem Browser zu `http://todoapi-${RHT_OCP4_DEV_USER}-template`.

Kapitel 7 | Bereitstellen von Anwendungen mit mehreren Containern

`${RHT_OCP4_WILDCARD_DOMAIN}/todo/`. Die Anwendung To Do List sollte angezeigt werden.

**Anmerkung**

Der abschließende Schrägstrich in der oben erwähnten URL ist notwendig. Wenn Sie dies in der URL nicht angeben, können Probleme mit der Anwendung auftreten.

The screenshot displays the 'To Do List Application' interface. On the left, there is a 'To Do List' section containing a table with two rows:

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

On the right, there is an 'Add Task' section with fields for 'Description' (containing 'Add Description.') and 'Completed' (with an unchecked checkbox). Below these 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'.

Abbildung 7.6: Anwendung „To Do List“

Beenden

Führen Sie auf workstation das Skript `lab multicontainer-openshift finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit ist die angeleitete Übung beendet.

► Praktische Übung

Bereitstellen von Anwendungen mit mehreren Containern

Ergebnisse

Sie sollten in der Lage sein, eine OpenShift-Anwendung mit mehreren Containern zu erstellen und über einen Webbrower darauf zuzugreifen.

Bevor Sie Beginnen

Öffnen Sie als Benutzer student auf workstation ein Terminal, und führen Sie die folgenden Befehle aus:

```
[student@workstation ~]$ lab multicontainer-review start  
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-review
```

Anweisungen

1. Melden Sie sich beim OpenShift-Cluster an, und erstellen Sie ein neues Projekt für diese Übung.
2. Erstellen Sie das Datenbankcontainer-Image, das sich im Verzeichnis `images/mysql` befindet, und veröffentlichen Sie es in Ihrem Quay.io-Repository.
3. Erstellen Sie das PHP-Container-Image unter `images/quote-php`, und veröffentlichen Sie es in Ihrem Quay.io-Repository.



Warnung

Beide Repositorys müssen in `quay.io` öffentlich sein, damit OpenShift die Images abrufen kann. Details zum Ändern der Repository-Sichtbarkeit finden Sie im Abschnitt **Sichtbarkeit von Repositorys** von *Anhang C, Erstellen eines Quay-Benutzerkontos*.

4. Wechseln Sie zum Verzeichnis `/home/student/D0180/labs/multicontainer-review/`, und überprüfen Sie die bereitgestellte Vorlage `quote-php-template.json`.
5. Laden Sie die PHP-Anwendungsvorlage hoch, damit sie von jedem Entwickler verwendet werden kann, der Zugriff auf Ihr Projekt hat.
6. Verarbeiten Sie die hochgeladene Vorlage und erstellen Sie die Anwendungsressourcen.
7. Stellen Sie den Service bereit.
Ermöglichen Sie den Zugriff auf die PHP Quote-Anwendung über den OpenShift-Router und die Erreichbarkeit über ein externes Netzwerk.
8. Testen Sie die Anwendung, und verifizieren Sie, dass eine aussagekräftige Meldung ausgegeben wird.

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem workstation-Rechner den Befehl `lab multicontainer-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf workstation den Befehl `lab multicontainer-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Bereitstellen von Anwendungen mit mehreren Containern

Ergebnisse

Sie sollten in der Lage sein, eine OpenShift-Anwendung mit mehreren Containern zu erstellen und über einen Webbrowser darauf zuzugreifen.

Bevor Sie Beginnen

Öffnen Sie als Benutzer student auf workstation ein Terminal, und führen Sie die folgenden Befehle aus:

```
[student@workstation ~]$ lab multicontainer-review start  
[student@workstation ~]$ cd ~/D0180/labs/multicontainer-review
```

Anweisungen

1. Melden Sie sich beim OpenShift-Cluster an, und erstellen Sie ein neues Projekt für diese Übung.
 - 1.1. Melden Sie sich auf dem Rechner workstation als der Benutzer an, der in der ersten Übung bereitgestellt wurde.

```
[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>
```

Wenn der Befehl `oc login` Sie bezüglich unsicherer Verbindungen zu einer Eingabe auffordert, antworten Sie mit „y“ (ja).

- 1.2. Erstellen Sie ein neues Projekt in OpenShift namens `deploy`, und stellen Sie diesem Ihren OpenShift-Benutzernamen voran:

```
[student@workstation multicontainer-review]$ oc new-project \  
> ${RHT_OCP4_DEV_USER}-deploy  
Now using project ...output omitted...
```

2. Erstellen Sie das Datenbankcontainer-Image, das sich im Verzeichnis `images/mysql` befindet, und veröffentlichen Sie es in Ihrem Quay.io-Repository.
 - 2.1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an.

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

- 2.2. Erstellen Sie das MySQL-Datenbank-Image unter Verwendung des bereitgestellten Containerfiles im Verzeichnis `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. Übertragen Sie das MySQL-Image per Push-Vorgang an Ihr Quay.io-Repository.

Kennzeichnen Sie das Image mit `quay.io/${RHT_OCP4_QUAY_USER}/do180-mysql-80-rhel8`, und übertragen Sie es per Push-Vorgang an die Registry `quay.io`, damit OpenShift es in der Vorlage verwenden kann. Damit Sie Images per Push-Vorgang an `quay.io` übertragen können, müssen Sie sich zunächst mit Ihren eigenen Anmeldedaten anmelden.

```
[student@workstation mysql]$ podman login quay.io -u ${RHT_OCP4_QUAY_USER}
Password: your_quay_password
Login Succeeded!
```

Führen Sie im Terminalfenster die folgenden Befehle aus, um das Image zu taggen (kennzeichnen) und zu übertragen.

```
[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
```

Kehren Sie zum vorherigen Verzeichnis zurück.

```
[student@workstation mysql]$ cd ~/DO180/labs/multicontainer-review
```

3. Erstellen Sie das PHP-Container-Image unter `images/quote-php`, und veröffentlichen Sie es in Ihrem Quay.io-Repository.

**Warnung**

Beide Repositorys müssen in `quay.io` öffentlich sein, damit OpenShift die Images abrufen kann. Details zum Ändern der Repository-Sichtbarkeit finden Sie im Abschnitt **Sichtbarkeit von Repositorys** von ???.

- 3.1. Erstellen Sie das PHP-Image unter Verwendung des bereitgestellten Containerfiles im Verzeichnis `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. Kennzeichnen und übertragen Sie das PHP-Image per Push-Vorgang an Ihre Quay.io-Registry.

Kennzeichnen Sie das Image mit `quay.io/${RHT_OCP4_QUAY_USER}/do180-quote-php`, und übertragen Sie es per Push-Vorgang an Quay.io, damit OpenShift es in der Vorlage verwenden kann.

```
[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. Wechseln Sie zum Verzeichnis `/home/student/D0180/labs/multicontainer-review/`, und überprüfen Sie die bereitgestellte Vorlage `quote-php-template.json`.
- 4.1. Beachten Sie die Definitionen und die Konfiguration der Pods und Services und die Anforderungen auf persistente Volumes, die in der Vorlage definiert sind.

```
[student@workstation quote-php]$ cd ~/D0180/labs/multicontainer-review
```

5. Laden Sie die PHP-Anwendungsvorlage hoch, damit sie von jedem Entwickler verwendet werden kann, der Zugriff auf Ihr Projekt hat.
- 5.1. Laden Sie mithilfe des Befehls `oc create -f` die Vorlagendatei in das Projekt hoch.

```
[student@workstation multicontainer-review]$ oc create -f quote-php-template.json
template.template.openshift.io/quote-php-persistent created
```

6. Verarbeiten Sie die hochgeladene Vorlage und erstellen Sie die Anwendungsressourcen.
- 6.1. Führen Sie den Befehl `oc process` aus, um die Vorlagendatei zu verarbeiten. Stellen Sie den Parameter `RHT_OCP4_QUAY_USER` mit dem Namespace `quay.io` bereit, in dem sich die Images befinden. Führen Sie den Befehl `pipe` aus, um das Ergebnis an den Befehl `oc create` zu senden und anhand der Vorlage eine Anwendung zu erstellen.

```
[student@workstation multicontainer-review]$ oc process quote-php-persistent \
> -p RHT_OCP4_QUAY_USER=${RHT_OCP4_QUAY_USER} \
> | oc create -f -
pod/mysql created
```

Kapitel 7 | Bereitstellen von Anwendungen mit mehreren Containern

```
pod/quote-php created
service/quote-php created
service/mysql created
persistentvolumeclaim/dbinit created
persistentvolumeclaim/dbclaim created
```

- 6.2. Überprüfen Sie den Status der Bereitstellung mithilfe des Befehls `oc get pods` und der Option `-w`, um den Bereitstellungsstatus zu überwachen. Warten Sie, bis beide Pods ausgeführt werden. Das Starten beider Pods kann einige Zeit in Anspruch nehmen.

```
[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
```

Drücken Sie Strg+C, um den Befehl zu verlassen.

- 7.** Stellen Sie den Service bereit.

Ermöglichen Sie den Zugriff auf die PHP Quote-Anwendung über den OpenShift-Router und die Erreichbarkeit über ein externes Netzwerk.

- 7.1. Verwenden Sie den Befehl `oc expose`, um den Service quote-php verfügbar zu machen.

```
[student@workstation multicontainer-review]$ oc expose svc quote-php
route.route.openshift.io/quote-php exposed
```

- 8.** Testen Sie die Anwendung, und verifizieren Sie, dass eine aussagekräftige Meldung ausgegeben wird.

- 8.1. Führen Sie den Befehl `oc get route` aus, um den FQDN zu suchen, in dem die Anwendung verfügbar ist. Beachten Sie den FQDN für die App.

Führen Sie im Terminalfenster den folgenden Befehl aus.

```
[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. Wechseln Sie zum Verzeichnis „/home/student“.

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

- 8.3. Überprüfen Sie mit dem Befehl `curl` die REST-API für die Anwendung 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.
```



Anmerkung

Der in der obigen Ausgabe angezeigte Text kann unterschiedlich sein, der curl-Befehl sollte jedoch erfolgreich ausgeführt werden.

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem workstation-Rechner den Befehl `lab multicontainer-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

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

Beenden

Führen Sie auf workstation den Befehl `lab multicontainer-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Software-defined Networks ermöglichen die Kommunikation zwischen Containern. Container müssen zum Kommunizieren an dasselbe Software-defined Network angehängt sein.
- Containerisierte Anwendungen können für die Suche nach Services keine festen IP-Adressen oder Hostnamen verwenden.
- Podman verwendet Container Network Interface (CNI), um ein Software-defined Network zu erstellen, und hängt alle Container auf dem Host an dieses Netzwerk an. Kubernetes und OpenShift erstellen ein Software-defined Network zwischen allen Containern in einem Pod.
- In demselben Projekt fügt Kubernetes für jeden Service eine Reihe von Variablen in alle Pods ein.
- Kubernetes-Vorlagen automatisieren die Erstellung von Anwendungen, die aus mehreren Ressourcen bestehen. Vorlagenparameter ermöglichen die Verwendung derselben Werte beim Erstellen mehrerer Ressourcen.

Kapitel 8

Fehlerbehebung bei containerisierten Anwendungen

Ziel

Fehlerbehebung bei containerisierten Anwendungen, die auf OpenShift bereitgestellt werden

Ziele

- Fehlerbehebung bei Anwendungsbuids und Bereitstellungen in OpenShift
- Implementierung der Methoden zur Fehlerbehebung und für das Debugging bei containerisierten Anwendungen

Abschnitte

- Fehlerbehebung bei S2I-Builds und - Bereitstellungen (und angeleitete Übung)
- Fehlerbehebung bei containerisierten Anwendungen (und angeleitete Übung)

Praktische Übung

- Fehlerbehebung bei containerisierten Anwendungen

Fehlerbehebung bei S2I-Builds und -Bereitstellungen

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Fehlerbehebung bei einer Anwendung anhand der Build- und Bereitstellungsschritte in OpenShift
- Analyse der OpenShift-Protokolle zur Identifizierung von Problemen während des Build- und Bereitstellungsprozesses

Einführung in den S2I-Prozess

Der Source-to-Image-Prozess (S2I) ist eine einfache Methode, um Images basierend auf Programmiersprachen des Anwendungsquellcodes in OpenShift automatisch zu erstellen. Obwohl dieser Prozess oftmals eine praktische Möglichkeit bietet, um Anwendungen bereitzustellen, können während der S2I-Image-Erstellung Probleme auftreten, entweder durch die Eigenschaften der Programmiersprache oder durch die Laufzeitumgebung, die eine Zusammenarbeit von Entwicklern und Administratoren erfordern.

Es ist wichtig, sich mit dem grundlegenden Workflow der meisten Programmiersprachen vertraut zu machen, die von OpenShift unterstützt werden. Der S2I-Prozess zur Image-Erstellung besteht aus zwei wesentlichen Schritten:

- Build-Prozess: Kompiliert den Quellcode, lädt Library-Abhängigkeiten herunter und packt die Anwendung als ein Container-Image. Im Rahmen des Build-Prozesses wird das Image außerdem für den Bereitstellungsschritt an die OpenShift-Registry übertragen. Die **BuildConfig**-Ressourcen (BC) von OpenShift steuern den Build-Schritt.
- Bereitstellungsprozess: Zuständig für das Starten eines Pods und die Bereitstellung der Anwendung für OpenShift. Dieser Schritt wird nach dem Build-Prozess ausgeführt, jedoch nur dann, wenn der Build-Schritt erfolgreich war. Die **Deployment**-Ressourcen von OpenShift steuern den Bereitstellungsschritt.

Für den S2I-Prozess verwendet jede Anwendung ihre eigenen **BuildConfig**- und **Deployment**-Objekte, deren Name mit dem Anwendungsnamen übereinstimmt. Der Bereitstellungsprozess wird abgebrochen, wenn der Build-Prozess fehlschlägt.

Der S2I-Prozess startet jeden Schritt in einem separaten Pod. Der Build-Prozess erstellt einen Pod mit dem Namen <application-name>-build-<number>-<string>. Für jeden Build-Versuch wird der gesamte Build-Schritt ausgeführt und ein Protokoll gespeichert. Nach einem erfolgreichen Build wird die Anwendung in einem separaten Pod namens <application-name>-<string> gestartet.

Auf die Details für jeden Schritt kann über die OpenShift Web Console zugegriffen werden. Um Build-Probleme zu identifizieren, können die Protokolle für ein Build ausgewertet und analysiert werden, indem Sie im linken Bereich auf den Link **Builds** klicken, der wie folgt dargestellt wird.

Name	Namespace	Status	Created
httpd-example-1	troubleshoot	Complete	2 minutes ago

Abbildung 8.1: Erstellen von Instanzen eines Projekts

Für jeden Build-Versuch wird ein Verlauf des Builds, der mit einer Nummer versehen ist, zur Auswertung bereitgestellt. Durch Klicken auf den Build-Namen wird die Detailseite des Builds geöffnet:

Name	Namespace	Labels	Status	Type	Git Repository
httpd-example-1	troubleshoot	app=httpd-example, buildconfig=httpd-example, openshift.io/build-config.name=httpd-example, openshift.io/build.start-policy=Serial	Complete	Source	https://github.com/sclorg/httpd-ex.git

Abbildung 8.2: Detailansicht einer Build-Instanz

Für jeden Build-Versuch wird ein Verlauf des Builds, der mit einer Nummer versehen ist, zur Auswertung bereitgestellt. Durch Klicken auf den Build-Namen wird die Detailseite des Builds geöffnet.

Die Registerkarte **Logs** der Build-Detailseite zeigt die von der Build-Ausführung generierte Ausgabe an. Diese Protokolle sind hilfreich, um Build-Probleme zu identifizieren.

Verwenden Sie den Link **Deployment** unter dem Abschnitt **Workloads** im linken Bereich, um Probleme während des Bereitstellungsprozesses zu identifizieren.

Nachdem Sie das entsprechende Bereitstellungsobjekt ausgewählt haben, werden die Details im Abschnitt **Details** angezeigt.

Die Befehlszeilschnittstelle oc verfügt über mehrere Unterbefehle zum Verwalten der Protokolle. Ähnlich wie in der Webschnittstelle steht eine Reihe von Befehlen zur Verfügung, die Informationen zu den jeweiligen Schritten bereitstellen. Um beispielsweise die Protokolle von einer Build-Konfiguration abzurufen, führen Sie den folgenden Befehl aus:

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

Führen Sie bei einem fehlgeschlagenen Build nach dem Auffinden und Korrigieren der Probleme den folgenden Befehl aus, um einen neuen Build anzufordern:

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

Durch diesen Befehl wird von OpenShift automatisch ein neuer Pod mit dem Build-Prozess erzeugt.

Bereitstellungsprotokolle können mit dem Befehl oc überprüft werden:

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

Wenn die Bereitstellung ausgeführt wird oder fehlgeschlagen ist, gibt der Befehl die Protokolle des Prozessbereitstellungsprozesses zurück. Andernfalls gibt der Befehl die Protokolle aus dem Pod der Anwendung zurück.

Beschreibung häufiger Probleme

Manchmal erfordert der Quellcode einige Anpassungen, die in containerisierten Umgebungen möglicherweise nicht verfügbar sind, z. B. Datenbankanmeldeinformationen, Dateisystemzugriff oder Nachrichtenwarteschlangeninformationen. Diese Werte haben normalerweise die Form von internen Umgebungsvariablen. Entwickler, die den S2I-Prozess verwenden, müssen möglicherweise auf diese Informationen zugreifen.

Der Befehl oc logs stellt im Rahmen der Ausführung eines Pods verschiedene wichtige Informationen über Build-, Bereitstellungs- und Ausführungsprozesse einer Anwendung bereit. Die Protokolle zeigen möglicherweise fehlende Werte oder Optionen an, die aktiviert werden müssen, sowie fehlerhafte Parameter, Flags oder Umgebungsinkompatibilitäten.



Anmerkung

Anwendungsprotokolle müssen eindeutig bezeichnet werden, damit Probleme schnell identifiziert werden können, ohne die internen Container-Informationen lesen zu müssen.

Fehlerbehebung bei Berechtigungsproblemen

OpenShift führt S2I-Container aus, welche Red Hat Enterprise Linux als Basis-Image verwenden, und jede Laufzeitdifferenz kann dazu führen, dass der S2I-Prozess fehlschlägt. Manchmal stoßen Entwickler auf Berechtigungsprobleme, z. B. wenn ein Zugriff aufgrund falscher Berechtigungen verweigert wurde oder falsche Umgebungsberechtigungen von Administratoren festgelegt wurden. S2I-Images erzwingen die Verwendung eines anderen Benutzers als der Root-Benutzer, um auf Dateisysteme und externe Ressourcen zuzugreifen. Darüber hinaus erzwingt Red Hat Enterprise Linux 7 SELinux-Richtlinien, die den Zugriff auf bestimmte Dateisystemressourcen, Netzwerkports oder Prozesse beschränken.

Einige Container können eine spezifische Benutzer-ID erfordern. S2I ist jedoch so konzipiert, dass Container gemäß der Standardsicherheitsrichtlinie von OpenShift unter Verwendung eines zufälligen Benutzers ausgeführt werden.

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

Das folgende Dockerfile erstellt einen Nexus-Container. Beachten Sie die USER-Anweisung, die angibt, dass der `nexus`-Benutzer verwendet werden sollte:

```
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...
```

Der Versuch, das von diesem Dockerfile generierte Image ohne entsprechende Volume-Berechtigungen zu verwenden, führt beim Container-Start zu Fehlern:

```
$ 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...
```

Um dieses Problem zu lösen, lockern Sie die Sicherheit des OpenShift-Projekts mit dem Befehl `oc adm policy`:

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

Durch den Befehl `oc adm policy` kann OpenShift Container-Prozesse mit Nicht-Root-Benutzern ausführen. Die im Container verwendeten Dateisysteme müssen jedoch auch für den laufenden Benutzer verfügbar sein. Dies ist besonders wichtig, wenn der Container Volume-Mounts enthält.

Um Dateisystemberechtigungsprobleme zu vermeiden, müssen lokale Ordner, die für das Bereitstellen von Container-Volumes verwendet werden, die folgenden Voraussetzungen erfüllen:

- Der Benutzer, der die Container-Prozesse ausführt, muss der Besitzer des Ordners sein oder über die erforderlichen Rechte verfügen. Führen Sie den Befehl `chown` aus, um die Ordnerinhaberschaft zu aktualisieren.
- Der lokale Ordner muss die SELinux-Anforderungen erfüllen, um als Container-Volume verwendet zu werden. Weisen Sie dem Ordner die Gruppe `container_file_t` zu, indem Sie den Befehl `semanage fcontext -a -t container_file_t <folder>` ausführen. Aktualisieren Sie anschließend mithilfe des Befehls `restorecon -R <folder>` die Berechtigungen.

Fehlerbehebung bei ungültigen Parametern

Anwendungen mit mehreren Containern können Parameter, zum Beispiel Anmeldeinformationen, gemeinsam nutzen. Stellen Sie sicher, dass dieselben Werte für Parameter alle Container in der Anwendung erreichen. Stellen Sie beispielsweise für eine Python-Anwendung, die in einem Container ausgeführt wird und mit einem anderen Container verbunden ist, der eine Datenbank

ausführt, sicher, dass die beiden Container denselben Benutzernamen und dasselbe Kennwort für die Datenbank verwenden. Normalerweise wird in den Protokollen aus dem Anwendungs-Pod genau angegeben, welche Probleme vorliegen und wie diese gelöst werden.

Bei der Zentralisierung freigegebener Parameter empfiehlt es sich, sie in ConfigMaps zu speichern. Diese ConfigMaps können über Deployment als Umgebungsvariablen in Container injiziert werden. Durch das Injizieren derselben ConfigMap in unterschiedliche Container wird sichergestellt, dass nicht nur die gleichen Umgebungsvariablen, sondern auch die gleichen Werte verfügbar sind. Beachten Sie die folgende Pod-Ressourcendefinition:

```
apiVersion: v1
kind: Pod
...output omitted...
spec:
  containers:
    - name: test-container
  ...output omitted...
  env:
    - name: ENV_1 ①
      valueFrom:
        configMapKeyRef:
          name: configMap_name1
          key: configMap_key_1
  ...output omitted...
  envFrom:
    - configMapRef:
      name: configMap_name_2 ②
  ...output omitted...
```

- ① Eine ENV_1-Umgebungsvariable wird in den Container eingefügt. Ihr Wert ist der Wert für den Eintrag configMap_key_1 in der configMap configMap_name1.
- ② Alle Einträge in configMap_name_2 werden als Umgebungsvariablen mit demselben Namen und gleichen Werten in den Container injiziert.

Fehlerbehebung bei Problemen mit der Volumenmontage

Bei der erneuten Bereitstellung einer Anwendung, die auf einem lokalen Dateisystem ein persistentes Volume verwendet, können Pods Probleme bei der Zuweisung einer Anforderung für ein persistentes Volume haben, auch wenn das persistente Volume angibt, dass die Anforderung freigegeben wurde.

Um das Problem zu beheben, löschen Sie die Anforderung für das persistente Volume und dann das persistente Volume. Erstellen Sie dann das persistente Volume neu:

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

Fehlerbehebung bei veralteten Images

OpenShift ruft Images aus der in einem Image-Stream angegebenen Quelle ab, es sei denn, OpenShift findet ein lokal zwischengespeichertes Image auf dem Knoten, auf dem der Pod ausgeführt werden soll. Wenn Sie ein neues Image unter Verwendung desselben Namens und Tags

in die Registry übertragen, müssen Sie das Image mit dem Befehl `podman rmi` auf jedem Knoten entfernen, auf dem der Pod ausgeführt werden soll.

Führen Sie den Befehl `oc adm prune` aus, um veraltete Images und andere Ressourcen automatisch zu entfernen.



Literaturhinweise

Weitere Informationen zur Fehlerbehebung bei Images finden Sie im Abschnitt *Images* in der Dokumentation zu OpenShift Container Platform unter:

Creating Images

https://docs.openshift.com/container-platform/4.6/openshift_images/create-images.html

Informationen zur Verwendung von ConfigMap zum Erstellen von Container-Umgebungsvariablen finden Sie unter *Consuming in Environment Variables* der

Configure a Pod to use ConfigMaps

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

► Angeleitete Übung

Fehlerbehebung bei OpenShift-Builds

In dieser Übung beheben Sie die in einem Build- und Bereitstellungsprozess von OpenShift auftretenden Probleme.

Ergebnisse

Sie sollten in der Lage sein, die während des Build- und Bereitstellungsprozesses einer Node.js-Anwendung auftretenden Probleme zu identifizieren und zu lösen.

Bevor Sie Beginnen

Auf einen aktiven OpenShift-Cluster

Rufen Sie die Lab-Dateien ab und überprüfen Sie, ob Docker und der OpenShift-Cluster ausgeführt werden, indem Sie den folgenden Befehl ausführen.

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

Anweisungen

- 1. Laden Sie die Konfiguration Ihrer Kursumgebung. Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

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

- 2. Wechseln Sie zu Ihrem lokalen Klon des Git-Repositorys D0180-apps, und checken Sie den Branch master des Kurs-Repositorys aus, um sicherzustellen, dass Sie diese Übung von einem als fehlerfrei bekannten Zustand aus starten:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

- 3. Erstellen Sie einen neuen Branch, um alle Änderungen zu speichern, die Sie während dieser Übung vornehmen:

```
[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. Melden Sie sich mit dem konfigurierten Benutzer, Passwort und der Master-API-URL bei OpenShift an.

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

Erstellen Sie ein neues Projekt mit dem Namen *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. Erstellen Sie die neue Node.js-Anwendung mit dem Image Hello World aus dem Verzeichnis nodejs-helloworld unter <https://github.com/yourgituser/D0180-apps/>.
- 5.1. Führen Sie den Befehl `oc new-app` zum Erstellen der Node.js-Anwendung aus. Der Befehl wird in der Datei `~/D0180/labs/troubleshoot-s2i/command.txt` bereitgestellt.

```
[student@workstation D0180-apps]$ oc new-app \
> --context-dir=nodejs-helloworld \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#troubleshoot-s2i \
> -i nodejs:12 --name nodejs-hello --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/npm-proxy
--> Found image e9b0166 ...output omitted...

      Node.js 12
...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.
```

-i gibt das zu verwendende Builder-Image an, in diesem Fall `nodejs:12`.

Die Option `--context-dir` legt fest, welcher Ordner im Projekt den Quellcode der zu erstellenden Anwendung enthält.

Die Option `--build-env` definiert eine Umgebungsvariable für den Builder-Pod. In diesem Fall wird dem Builder-Pod die Umgebungsvariable `npm_config_registry` bereitgestellt, sodass die NPM-Registry erreicht werden kann.



Wichtig

Im vorherigen Befehl dürfen keine Leerzeichen zwischen `registry=` und der URL des Nexus-Servers vorhanden sein.

- 5.2. Warten Sie, bis die Anwendung den Erstellungsvorgang beendet hat, indem Sie den Fortschritt mithilfe des Befehls `oc get pods -w` überwachen. Der Pod wechselt vom Status `running` zu `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
```

Der Build-Prozess schlägt fehl, aus diesem Grund wird keine Anwendung ausgeführt. Fehlgeschlagene Build-Prozesse sind in der Regel die Folgen von Syntaxfehlern im Quellcode oder fehlenden Abhängigkeiten. Im nächsten Schritt werden die spezifischen Ursachen für den fehlgeschlagenen Prozess untersucht.

- 5.3. Evaluieren Sie die Fehler, die während des Build-Prozesses auftreten.

Das Build wird durch die Build-Konfiguration (`bc`) ausgelöst, die von OpenShift beim Starten des S2I-Prozesses erstellt wurde. Standardmäßig erstellt der S2I-Prozess von OpenShift eine Build-Konfiguration namens `nodejs-hello`, die für die Auslösung des Build-Prozesses zuständig ist.

Führen Sie den Befehl `oc` mit dem Unterbefehl `logs` in einem Terminalfenster aus, um die Ausgabe des Build-Prozesses zu überprüfen:

```
[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...
```

Das Protokoll zeigt während des Build-Prozesses einen Fehler an. Diese Ausgabe zeigt an, dass es keine kompatible Version für die `express`-Abhängigkeit gibt. Dies liegt daran, dass das von der Express-Abhängigkeit verwendete Format ungültig ist.

► **6.** Aktualisieren Sie den Buildprozess für das Projekt.

Der Entwickler verwendet eine nicht standardmäßige Version des Express-Frameworks, das lokal auf jeder Entwickler-Workstation verfügbar ist. Den Unternehmensstandards entsprechend muss die Version von der offiziellen Node.js-Registry heruntergeladen werden. Diese ist aus Entwicklersicht mit der Version 4.14.x kompatibel.

6.1. Korrigieren Sie die Datei `package.json`.

Öffnen Sie in Ihrem bevorzugten Editor die Datei `~/D0180-apps/nodejs-helloworld/package.json`. Überprüfen Sie die von den Entwicklern bereitgestellten Abhängigkeiten. Es verwendet eine falsche Version der Express-Abhängigkeit, die mit der vom Unternehmen bereitgestellten Version (~4.14.2) nicht kompatibel ist. Aktualisieren Sie die Abhängigkeit wie folgt.

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



Anmerkung

Beachten Sie das `x` in der Version. Es gibt an, dass die höchste Version verwendet werden sollte. Die Version muss jedoch mit `4.14` beginnen.

6.2. Übernehmen Sie die am Projekt vorgenommenen Änderungen und übertragen Sie sie.

Führen Sie im Terminalfenster den folgenden Befehl aus, um die Änderungen zu übernehmen und zu übertragen:

```
[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.** Starten Sie den S2I Prozess erneut.

7.1. Führen Sie den folgenden Befehl aus, um den Build-Prozess neu zu starten:

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

Der Build-Prozess wird neu gestartet und ein neuer Build-Pod wird erstellt. Überprüfen Sie das Protokoll, indem Sie den Befehl `oc logs` ausführen.

```
[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
```

Der Build ist erfolgreich, dies bedeutet jedoch nicht, dass die Anwendung gestartet wurde.

- 7.2. Überprüfen Sie den Status des aktuellen Build-Prozesses. Führen Sie den Befehl `oc get pods` aus, um den Status der Node.js-Anwendung zu überprüfen.

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

Gemäß der folgenden Ausgabe ist der zweite Build abgeschlossen, aber die Anwendung befindet sich im Fehlerzustand.

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

Der Name des Anwendungs-Pods (`nodejs-hello-7b99966464-fw4r8`) wird zufällig generiert und kann von Ihrem abweichen.

- 7.3. Überprüfen Sie die vom Anwendungs-Pod generierten Protokolle.

```
[student@workstation D0180-apps]$ oc logs nodejs-hello-7b99966464-fw4r8
...output omitted...
npm info using npm@6.14.8
npm info using node@v12.19.1
npm ERR! missing script: start
...output omitted...
```



Anmerkung

Der Befehl `oc logs nodejs-hello-7b99966464-fw4r8` speichert die Protokolle vom Bereitstellungs-Pod. Bei einer erfolgreichen Bereitstellung werden mit diesem Befehl die Protokolle aus dem Anwendungs-Pod ausgegeben, wie zuvor gezeigt.

Wenn in den Bereitstellungsprotokollen der Fehler nicht angezeigt wird, überprüfen Sie die Protokolle des Anwendungs-Pods, indem Sie den Befehl `oc logs Pod` ausführen. Ersetzen Sie hierbei „POD“ durch den Namen des Pods, der den Status `CrashLoopBackOff` hat.

Die Anwendung kann nicht gestartet werden, da die Deklaration zum Starten des Skripts fehlt.

- 8. Beheben Sie das Problem, indem Sie den Anwendungscode aktualisieren.

- 8.1. Aktualisieren Sie die Datei `package.json`, um einen Startbefehl zu definieren.

Die vorherige Ausgabe zeigt, dass der Datei `~/D0180-apps/nodejs-helloworld/package.json` das Attribut `start` im Feld `scripts` fehlt. Das Attribut `start` definiert einen Befehl, der beim Starten der Anwendung ausgeführt werden soll. Es ruft die Binärdatei `node` auf, welche die Anwendung `app.js` ausführt. Um das Problem zu beheben, fügen Sie das folgende Attribut zur Datei `package.json` hinzu. Vergessen Sie das Komma nach der Klammer nicht.

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

- 8.2. Übernehmen Sie die am Projekt vorgenommenen Änderungen und übertragen Sie sie:

```
[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
```

Fahren Sie mit der Bereitstellung aus dem S2I-Prozess fort.

- 8.3. Starten Sie den Build-Schritt neu.

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

- 8.4. Überprüfen Sie den Status des aktuellen Build-Prozesses. Führen Sie den Befehl aus, um den Status der Node.js-Anwendung abzurufen. Warten Sie, bis der letzte Build-Prozess abgeschlossen ist.

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

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

Entsprechend der Ausgabe ist der Build erfolgreich, und die Anwendung kann ohne Fehler starten. Anhand der Ausgabe wird auch ersichtlich, wie der Bereitstellungs-Pod (`nodejs-hello-3-build`) erstellt wurde und dass er erfolgreich abgeschlossen und beendet wurde. Da der neue Anwendungs-Pod (`nodejs-hello-7b99966464-dvtcz`) verfügbar ist, wird der alte (`nodejs-hello-7b99966464-fw4r8`) bereitgestellt.

- 8.5. Überprüfen Sie die vom Anwendungs-Pod `nodejs-hello` generierten Protokolle.

```
[student@workstation D0180-apps]$ oc logs nodejs-hello-6cd88b7b7-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@6.14.8
npm info using node@v12.19.1
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 /opt/app-root/src
> node app.js

Example app listening on port 8080!
```

Die Anwendung läuft jetzt auf Port 8080.

► 9. Testen Sie die Anwendung

- 9.1. Führen Sie den Befehl `oc` mit dem Unterbefehl `expose` aus, um die Anwendung verfügbar zu machen:

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

- 9.2. Rufen Sie die Adresse ab, die der Anwendung zugeordnet ist.

```
[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. Greifen Sie mit dem Befehl `curl` über die `workstation`-VM auf die Anwendung zu:

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

Die Ausgabe zeigt, dass die Anwendung ausgeführt wird.

Beenden

Führen Sie auf `workstation` das Skript `lab troubleshoot-s2i finish` aus, um diese Übung zu beenden.

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

Hiermit ist diese Übung beendet.

Fehlerbehebung bei containerisierten Anwendungen

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Implementierung der Methoden zur Fehlerbehebung und für das Debugging bei containerisierten Anwendungen
- Verwendung der Funktion zur Portweiterleitung des Client-Tools von OpenShift
- Anzeige der Container-Protokolle
- Anzeige von OpenShift-Cluster-Ereignissen

Weiterleitung von Ports zur Fehlerbehebung

Gelegentlich benötigen Entwickler und Systemadministratoren einen speziellen Netzwerkzugriff auf einen Container, der von Anwendungsbewutzern nicht benötigt wird. Beispielsweise müssen Entwickler möglicherweise die Verwaltungskonsole für einen Datenbank- oder Messaging-Service verwenden. Alternativ können Systemadministratoren per SSH auf einen Container zugreifen, um einen beendeten Service neu zu starten. Ein solcher Netzwerkzugriff in Form von Netzwerkports wird normalerweise nicht von den Standard-Container-Konfigurationen offen gelegt und erfordert in der Regel spezielle Clients, die von Entwicklern und Systemadministratoren verwendet werden.

Podman stellt Portweiterleitungsfeatures bereit, indem die Option `-p` zusammen mit dem Unterbefehl `run` verwendet wird. In diesem Fall besteht kein Unterschied zwischen dem Netzwerkzugriff für den herkömmlichen Anwendungszugriff und für die Fehlerbehebung. Zur Erinnerung sehen Sie nachfolgend ein Beispiel für die Konfiguration der Portweiterleitung durch Zuordnung des Ports vom Host zu einem Datenbankserver, der in einem Container ausgeführt wird:

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

Der vorherige Befehl ordnet den Host-Port 30306 dem Port 3306 im db-Container zu. Dieser Container wird aus dem `mysql`-Image erstellt, das einen MySQL-Server startet, der Port 3306 überwacht.

OpenShift stellt den Befehl `oc port-forward` bereit, um einen lokalen Port zu einem Pod-Port weiterzuleiten. Dies unterscheidet sich vom Zugriff auf einen Pod über eine Serviceressource:

- Die Zuordnung der Portweiterleitung wird nur auf der Workstation vorgenommen, in der der `oc`-Client ausgeführt wird, wohingegen ein Service einen Port für alle Netzwerkbewutzer zuweist.
- Während Services das Load Balancing für Verbindungen zu (potenziell) mehreren Pods vornehmen, werden bei einer Zuordnung von Portweiterleitungen Verbindungen an einen einzelnen Pod weitergeleitet.

Hier ist ein Beispiel für den Befehl `oc port-forward`:

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

Der vorherige Befehl leitet Port 30306 vom Entwicklercomputer an Port 3306 im db-Pod weiter, wo ein MySQL-Server (in einem Container) Netzwerkverbindungen akzeptiert.



Anmerkung

Stellen Sie sicher, dass das Terminalfenster ausgeführt wird, wenn Sie diesen Befehl ausführen. Das Schließen des Fensters oder das Abbrechen des Prozesses beendet das Port-Mapping.

Während der Befehl `podman run -p` für die Zuordnungsmethode (Portweiterleitungen) nur beim Starten des Containers konfiguriert werden kann, kann die Zuordnung über den Befehl `oc port-forward` nach Erstellung eines Pods jederzeit erstellt oder gelöscht werden.



Anmerkung

Die Erstellung eines NodePort-Servicetyps für einen Datenbank-Pod ist mit der Ausführung des Befehls `podman run -p` vergleichbar. Red Hat rät jedoch von der Verwendung des NodePort-Ansatzes ab, um zu vermeiden, dass der Service direkten Verbindungen ausgesetzt wird. Die Zuordnung über die Portweiterleitung in OpenShift gilt als sicherere Alternative.

Aktivieren von Remote-Debugging mit Portweiterleitung

Die Funktion zur Portweiterleitung kann auch für das externe Debugging verwendet werden. Viele integrierte Entwicklungsumgebungen (Integrated Development Environments, IDEs) bieten eine Option zum externen Debugging einer Anwendung.

Beispiel: JBoss Developer Studio (JBDS) ermöglicht Benutzern die Verwendung von Java Debug Wire Protocol (JDWP) zur Kommunikation zwischen einem Debugger (JBDS) und dem virtuellen Java-Rechner. Wenn diese Option aktiviert ist, können Benutzer jede Zeile eines Codes so durchgehen, wie er in Echtzeit ausgeführt werden würde.

Damit JDWP ordnungsgemäß funktioniert, muss Java Virtual Machine (JVM), auf dem die Anwendung durchgeführt wird, mit den Optionen zur Aktivierung der Remote-Debugging-Funktion gestartet werden. Beispielsweise müssen WildFly-Benutzer und JBoss EAP-Benutzer beim Starten des Anwendungsservers diese Optionen konfigurieren. In der folgenden Zeile der Datei `standalone.conf` wird das externe Debugging aktiviert, indem der JDWP TCP-Port 8787 für eine WildFly- oder EAP-Instanz geöffnet wird, die im Standalone-Modus ausgeführt wird:

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

Wenn der Server mit dem Debugger beginnt, der Port 8787 überwacht, muss eine Portweiterleitungszuordnung erstellt werden, um Verbindungen von einem lokalen nicht verwendeten TCP-Port zu Port 8787 im EAP-Pod weiterzuleiten. Wenn auf der Entwickler-Workstation kein lokaler JVM mit einer aktivierte externen Debugging-Funktion aktiviert ist, kann als lokaler Port auch Port 8787 verwendet werden.

Beim folgenden Befehl wird davon ausgegangen, dass ein WildFly-Pod namens `jappserver` einen Container über ein zuvor konfiguriertes Image ausführt, um das externe Debugging zu aktivieren:

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

Sobald der Debugger aktiviert wurde und die Zuordnung von Portweiterleitungen erstellt wird, können Benutzer in der IDE ihrer Wahl Haltepunkte festsetzen und den Debugger ausführen, indem sie auf den Hostnamen und den Debug-Port der Anwendung (in diesem Fall 8787) verweisen.

Zugriff auf Container-Protokolle

Podman und OpenShift bieten die Möglichkeit zur Anzeige von Protokollen in ausgeführten Containern und Pods, um die Fehlersuche zu erleichtern. Keiner von beiden kennt jedoch die anwendungsspezifischen Protokolle. Beide erwarten, dass die Anwendung so konfiguriert ist, dass alle Protokollausgaben zur Standardausgabe gesendet werden.

Aus Sicht des Host-Betriebssystems ist ein Container nur eine Prozessstruktur. Wenn Podman einen Container entweder direkt oder in einem RHOC-Cluster startet, werden die Standardausgabe des Containers und die Standardfehler umgeleitet, indem sie als Teil des temporären Storage des Containers auf dem Laufwerk gespeichert werden. Auf diese Weise können Container-Protokolle mithilfe der Befehle podman und oc angezeigt werden, auch nachdem der Container beendet wurde. Dies ist jedoch nicht möglich, wenn der Container entfernt wurde.

Führen Sie den folgenden podman-Befehl aus, um die Ausgabe eines ausgeführten Containers abzurufen:

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

In OpenShift gibt der folgende Befehl die Ausgabe für einen Container in einem Pod zurück:

```
$ oc logs <podName> [-c <containerName>]
```



Anmerkung

Der Container-Name ist optional, wenn nur ein einziger laufender Container vorhanden ist, da bei Ausführung des Befehls oc dieser Container als Standard-Container festgelegt und die Ausgabe zurückgegeben wird.

OpenShift-Ereignisse

Einige Entwickler halten Podman- und OpenShift-Protokolle für zu detailliert, was die Fehlersuche erschwert. Glücklicherweise bietet OpenShift eine hochwertige Protokoll- und Prüffunktion namens events.

OpenShift-Ereignisse signalisieren wichtige Aktionen wie das Starten eines Containers oder die Entfernung eines Pods.

Verwenden Sie zum Lesen von OpenShift-Ereignissen den Unterbefehl get mit dem events-Ressourcentyp für den Befehl oc wie folgt:

```
$ oc get events
```

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

Ereignisse, die über den oc-Befehl auf diese Weise aufgelistet werden, sind nicht gefiltert und erstrecken sich über den gesamten RHOC-Cluster. Die Verwendung einer Pipe zu Standard-UNIX-Filtern wie grep kann hilfreich sein. OpenShift bietet jedoch eine Alternative, um Cluster-Ereignisse zu untersuchen. Der Ansatz wird durch den Unterbefehl `describe` bereitgestellt.

Um beispielsweise nur die Ereignisse abzurufen, die sich auf einen mysql-Pod beziehen, verweisen Sie im Feld Ereignisse auf die Ausgabe des Befehls `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...
```

Zugriff auf ausgeführte Container

Die Befehle `podman logs` und `oc logs` können zum Anzeigen der von einem Container gesendeten Ausgabe nützlich sein. Die Ausgabe zeigt jedoch nicht unbedingt alle verfügbaren Informationen an, wenn die Anwendung zum Senden von Protokollen an eine Datei konfiguriert ist. Es kann vorkommen, dass in anderen Fehlerbehebungsszenarios die Containerumgebung so überprüft werden muss, wie sie von Prozessen innerhalb des Containers erkannt wird, z. B. die Überprüfung der externen Konnektivität.

Als Lösung stellen Podman und OpenShift den Unterbefehl `exec` bereit, der die Erstellung neuer Prozesse innerhalb eines laufenden Containers ermöglicht und die Standardausgabe und -eingabe dieser Prozesse an das Benutzerterminal umleitet. Auf dem folgenden Bildschirm wird der Befehl `podman exec` verwendet:

```
$ podman exec [options] container command [arguments]
```

Die allgemeine Syntax für den Befehl `oc exec` lautet:

```
$ oc exec [options] pod [-c container] -- command [arguments]
```

Um einen einzelnen interaktiven Befehl auszuführen oder eine Shell zu starten, fügen Sie die `-it`-Optionen hinzu. Im folgenden Beispiel wird eine Bash-Shell für den `myhttpd`-Pod gestartet:

```
$ oc exec -it myhttpd /bin/bash
```

Sie können mithilfe dieses Befehls auf Anwendungsprotokolle zugreifen, die auf der Festplatte (als Teil des temporären Storage des Containers) gespeichert sind. Um beispielsweise das Apache-Fehlerprotokoll von einem Container aus anzuzeigen, führen Sie den folgenden Befehl aus:

```
$ podman exec apache-container cat /var/log/httpd/error_log
```

Überschreiben von Container-Binärdateien

Viele Container-Images enthalten nicht alle Befehle zur Fehlerbehebung, die Benutzer in regulären Betriebssysteminstallationen erwarten, wie beispielsweise `telnet`, `netcat`, `ip` oder `traceroute`. Durch das Entfernen des Image von grundlegenden Dienstprogrammen oder Binärdateien bleibt das Image schlank und es werden viele Container pro Host ausgeführt.

Eine Möglichkeit, um vorübergehend auf einige dieser fehlenden Befehle zuzugreifen, besteht in der Bereitstellung der Host-Binärdateien, z. B. /bin, /sbin und /lib, als Volumes im Container. Dies ist möglich, da für die -v-Option aus dem Befehl `podman run` keine übereinstimmenden `VOLUME`-Anweisungen im Dockerfile des Container-Images vorhanden sein müssen.



Anmerkung

Um auf diese Befehle in OpenShift zugreifen zu können, müssen Sie die Definition der Pod-Ressource ändern, um die Objekte `volumeMounts` und `volumeClaims` zu definieren. Sie müssen auch ein persistentes `hostPath`-Volume erstellen.

Der folgende Befehl startet einen Container und überschreibt den Ordner `/bin` des Image mit dem des Hosts. Er startet auch eine interaktive Shell innerhalb des Containers:

```
$ podman run -it -v /bin:/bin image /bin/bash
```



Anmerkung

Das Verzeichnis der zu überschreibenden Binärdateien hängt vom Basis-BS-Image ab. Zum Beispiel erfordern einige Befehle gemeinsam genutzte Bibliotheken aus dem Verzeichnis `/lib`. Einige Linux-Distributionen haben unterschiedliche Inhalte in `/bin`, `/usr/bin`, `/lib` oder `/usr/lib`, was erfordert, dass die Option `-v` für jedes Verzeichnis verwendet wird.

Alternativ können Sie diese Dienstprogramme in das Basisimage aufnehmen. Fügen Sie dazu Anweisungen in eine Dockerfile-Build-Definition ein. Betrachten Sie zum Beispiel den folgenden Auszug einer Dockerfile-Definition, die dem in diesem Kurs verwendeten `rhel7.5`-Image untergeordnet ist. Die RUN-Anweisung installiert die Tools, die häufig zur Fehlerbehebung im Netzwerk verwendet werden:

```
FROM ubi7/ubi:7.7

RUN yum install -y \
    less \
    dig \
    ping \
    iputils && \
    yum clean all
```

Wenn das Image und der Container erstellt werden, sind sie außer in Bezug auf die zusätzlichen, verfügbaren Tools mit einem `rhel7.5`-Container-Image identisch.

Übertragen von Dateien in und aus Containern

Bei der Problembehandlung oder Verwaltung einer Anwendung müssen Sie möglicherweise Dateien von laufenden Containern abrufen oder übertragen, z. B. Konfigurationsdateien oder Protokolldateien. Es gibt mehrere Möglichkeiten, Dateien in und aus Containern zu verschieben, wie in der folgenden Liste beschrieben.

Volume-Bereitstellungen

Eine weitere Möglichkeit zum Kopieren von Dateien aus dem Host in einen Container ist die Verwendung von Volume-Bereitstellungen. Sie können ein lokales Verzeichnis bereitstellen,

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

um Daten in einen Container zu kopieren. Beispiel: Der folgende Befehl legt das Host-Verzeichnis `/conf` als Volume zur Verwendung mit dem Apache-Konfigurationsverzeichnis fest. Dies bietet eine bequeme Möglichkeit, den Apache-Server zu verwalten, ohne das Container-Image neu erstellen zu müssen.

```
$ podman run -v /conf:/etc/httpd/conf -d do180/apache
```

podman cp

Der Unterbefehl `cp` ermöglicht es Benutzern, Dateien in einen laufenden Container zu kopieren oder daraus zu entfernen. Führen Sie den folgenden Befehl aus, um eine Datei in einen Container mit dem Namen `todoapi` zu kopieren.

```
$ podman cp standalone.conf todoapi:/opt/jboss/standalone/conf/standalone.conf
```

Um eine Datei aus einem Container in den Host zu kopieren, kehren Sie die Reihenfolge des vorherigen Befehls um.

```
$ podman cp todoapi:/opt/jboss/standalone/conf/standalone.conf .
```

Der Befehl `podman cp` hat den Vorteil, dass bereits gestartete Container verwendet werden können. Bei der folgenden Alternative (Volume-Bereitstellungen) muss der Befehl zum Starten eines Containers geändert werden.

podman exec

Bei bereits ausgeführten Containern kann der Befehl `podman exec` als Pipe zur Weiterleitung von Dateien in und aus dem ausgeführten Container verwendet werden. Dazu werden Befehle angehängt, die im Container ausgeführt werden. Im folgenden Beispiel wird dargestellt, wie eine SQL-Datei in einen MySQL-Container eingefügt und ausgeführt wird:

```
$ podman exec -i <container> mysql -uroot -proot < /path/on/host/db.sql < db.sql
```

Mit dem gleichen Konzept ist es möglich, Daten aus einem laufenden Container abzurufen und auf dem Host-Rechner abzulegen. Ein nützliches Beispiel hierfür ist die Verwendung des Dienstprogramms `mysqldump`, das eine Sicherung der MySQL-Datenbank aus dem Container erstellt und sie auf dem Host ablegt.

```
$ 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
```

Im vorherigen Befehl werden die Container-Umgebungsvariablen verwendet, um eine Verbindung zum MySQL-Server herzustellen, um das `mysqldump`-Dienstprogramm auszuführen und die Ausgabe an eine Datei auf dem Host-Rechner weiterzuleiten. Es wird davon ausgegangen, dass das Container-Image das Dienstprogramm `mysqldump` bereitstellt. Es ist daher nicht notwendig, die MySQL-Administrationstools auf dem Host zu installieren.

Der Befehl `oc rsync` bietet – ähnlich wie der Befehl `podman cp` – eine Funktion für Container, die in OpenShift-Pods ausgeführt werden.



Literaturhinweise

Weitere Informationen zur Portweiterleitung finden Sie im Abschnitt *Port Forwarding* der Dokumentation zu OpenShift Container Platform unter **Architecture**

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/architecture/index/

Weitere Informationen zu den CLI-Befehlen für die Portweiterleitung finden Sie im Kapitel *Port Forwarding* der OpenShift Container Platform-Dokumentation unter **Developing Applications**

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.6/html-single/applications/index/

► Angeleitete Übung

Konfigurieren von Apache-Containerprotokollen für das Debugging

In dieser Übung konfigurieren Sie einen Apache HTTPD-Container, um Protokolle an `stdout` zu senden, und überprüfen anschließend die Podman-Protokolle und -Ereignisse.

Ergebnisse

Sie sollten in der Lage sein, einen HTTPD-Container von Apache zu konfigurieren, um Debug- Protokolle an `stdout` zu senden und diese mithilfe des Befehls `podman logs` anzuzeigen.

Bevor Sie Beginnen

Auf einen aktiven OpenShift-Cluster

Rufen Sie die Lab-Dateien ab und überprüfen Sie, ob Docker und der OpenShift-Cluster ausgeführt werden, indem Sie den folgenden Befehl ausführen.

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

Anweisungen

- 1. Konfigurieren Sie den Apache-Webserver so, dass er Protokollmeldungen an die Standardausgabe sendet und die Standardprotokollstufe aktualisiert.
 - 1.1. Die Standardprotokollstufe für das Apache HTTPD-Image lautet `warn`. Ändern Sie die Standardprotokollstufe für den zu debuggenden Container, und leiten Sie Protokollmeldungen an `stdout` um, indem Sie die Standardkonfigurationsdatei `httpd.conf` überschreiben. Erstellen Sie dazu ein benutzerdefiniertes Image von der `Workstation`-VM.
Überprüfen Sie kurz die benutzerdefinierte Datei `httpd.conf` unter `/home/student/D0180/labs/troubleshoot-container/conf/httpd.conf`.
 - Untersuchen Sie die Direktive `ErrorLog` in der Datei:

```
ErrorLog "/dev/stdout"
```

Die Direktive sendet die HTTPD-Fehlerprotokollmeldungen an die Standardausgabe des Containers.

- Untersuchen Sie die Direktive `LogLevel` in der Datei.

```
LogLevel debug
```

Die Richtlinie ändert die Standardprotokollevne zu `debug`.

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

- Untersuchen Sie die CustomLog-Richtlinie in der Datei:

```
CustomLog "/dev/stdout" common
```

Die Direktive leitet die HTTPD-Zugriffsprotokollmeldungen an die Standardausgabe des Containers um.

- 2. Erstellen Sie einen benutzerdefinierten Container, um eine aktualisierte Konfigurationsdatei im Container zu speichern.

- 2.1. Führen Sie im Terminalfenster die folgenden Befehle aus, um ein neues Image zu erstellen.

```
[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. Überprüfen Sie, ob das Image erstellt wird.

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

Das neue Image muss im lokalen Storage verfügbar sein.

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. Erstellen Sie einen neuen HTTPD-Container aus dem benutzerdefinierten Image.

```
[student@workstation ~]$ podman run \  
> --name troubleshoot-container -d \  
> -p 10080:80 troubleshoot-container  
4c8bb12815cc02f4eef0254632b7179bd5ce230d83373b49761b1ac41fc067a9
```

- 4. Überprüfen Sie die Protokollmeldungen und -Ereignisse des Containers.

- 4.1. Zeigen Sie die Debug-Protokollmeldungen aus dem Container mithilfe des Befehls `podman logs` an:

```
[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
```

Beachten Sie die Debug-Protokolle, die in der Standardausgabe verfügbar sind.

- 4.2. Öffnen Sie ein neues Terminal, und greifen Sie mit dem Befehl curl auf die Startseite des Webservers zu:

```
[student@workstation ~]$ curl http://127.0.0.1:10080
Hello from the httpd-parent container!
```

- 4.3. Überprüfen Sie die neuen Einträge im Protokoll. Zeigen Sie im Terminalfenster, in dem der Befehl podman logs ausgeführt wird, die neuen Einträge an.

```
...output omitted...
10.0.2.2 - - [31/Mar/2021:14:06:03 +0000] "GET / HTTP/1.1" 200 39
```

- 4.4. Stoppen Sie den Podman-Befehl mit Strg+C.

Beenden

Führen Sie auf workstation das Skript lab troubleshoot-container finish aus, um diese praktische Übung abzuschließen.

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

Hiermit ist die angeleitete Übung beendet.

► Praktische Übung

Fehlerbehebung bei containerisierten Anwendungen

Ergebnisse

Sie sollten in der Lage sein, die während des Build- und Bereitstellungsprozesses einer Node.js-Anwendung auftretenden Probleme zu identifizieren und zu lösen.

Bevor Sie Beginnen

Auf einen aktiven OpenShift-Cluster

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Laden Sie die Konfiguration Ihrer Kursumgebung. Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:
2. Wechseln Sie zu Ihrem lokalen Klon des Git-Repositorys `D0180-apps`, und checken Sie den Branch `master` des Kurs-Repositorys aus, um sicherzustellen, dass Sie diese Übung von einem als fehlerfrei bekannten Zustand aus starten:
3. Erstellen Sie einen neuen Branch, um alle Änderungen zu speichern, die Sie während dieser Übung vornehmen:
4. Melden Sie sich mit dem konfigurierten Benutzer, Passwort und der Master-API-URL bei OpenShift an.
5. Erstellen Sie ein neues Projekt mit dem Namen `youruser-nodejs-app`:
6. Erstellen Sie im OpenShift-Projekt `youruser-nodejs-app` eine neue Anwendung anhand des Quellcodes aus dem Verzeichnis `nodejs-app` im Git-Repository unter <https://github.com/yourgituser/D0180-apps>. Umgebungsvariable `npm_config_registry` verfügbar unter `http://${RHT_OCP4_NEXUS_SERVER}/repository/npm-proxy`. Benennen Sie die Anwendung `nodejs-dev`.
Der Build-Prozess für die Anwendung sollte fehlschlagen. Überwachen Sie den Build-Prozess, und identifizieren Sie den Build-Fehler.
7. Aktualisieren Sie die Version der Abhängigkeit `express` in der Datei `package.json` mit dem Wert `4.x`. Bestätigen und übertragen Sie die Änderungen an das Git-Repository.
8. Erstellen Sie die Anwendung neu. Verifizieren Sie, dass die Anwendung fehlerfrei erstellt wird.
9. Verifizieren Sie, dass die Anwendung aufgrund eines Laufzeitfehlers nicht ausgeführt wird. Überprüfen Sie die Protokolle, und identifizieren Sie das Problem.
10. Korrigieren Sie die Schreibweise der Abhängigkeit in der ersten Zeile der Datei `server.js`. Bestätigen und übertragen Sie die Änderungen an der Anwendung an das Git-Repository.

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

Erstellen Sie die Anwendung neu. Verifizieren Sie nach dem Erstellen der Anwendung, dass die Anwendung ausgeführt wird.

11. Erstellen Sie eine Route für die Anwendung, und testen Sie den Zugriff auf die Anwendung. Es sollte eine Fehlermeldung angezeigt werden. Überprüfen Sie die Protokolle, um den Fehler zu ermitteln.
12. Ersetzen Sie zum Korrigieren des Fehlers `process.environment` durch `process.env` in der Datei `server.js`. Bestätigen und übertragen Sie die an der Anwendung vorgenommenen Änderungen an das Git-Repository. Erstellen Sie die Anwendung neu. Verifizieren Sie bei der Bereitstellung der neuen Anwendung, dass die Anwendung beim Zugriff auf die Anwendungs-URL keine Fehler generiert.

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem `workstation`-Rechner den Befehl `lab troubleshoot-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

Beenden

Führen Sie auf `workstation` den Befehl `lab troubleshoot-review finish` aus, um diese praktische Übung abzuschließen.

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Fehlerbehebung bei containerisierten Anwendungen

Ergebnisse

Sie sollten in der Lage sein, die während des Build- und Bereitstellungsprozesses einer Node.js-Anwendung auftretenden Probleme zu identifizieren und zu lösen.

Bevor Sie Beginnen

Auf einen aktiven OpenShift-Cluster

Öffnen Sie als der Benutzer `student` auf `workstation` ein Terminal, und führen Sie den folgenden Befehl aus:

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

Anweisungen

1. Laden Sie die Konfiguration Ihrer Kursumgebung. Führen Sie den folgenden Befehl aus, um die in der ersten angeleiteten Übung erstellten Umgebungsvariablen zu laden:

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

2. Wechseln Sie zu Ihrem lokalen Klon des Git-Repositories `D0180-apps`, und checken Sie den Branch `master` des Kurs-Repositorys aus, um sicherzustellen, dass Sie diese Übung von einem als fehlerfrei bekannten Zustand aus starten:

```
[student@workstation ~]$ cd ~/D0180-apps
[student@workstation D0180-apps]$ git checkout master
...output omitted...
```

3. Erstellen Sie einen neuen Branch, um alle Änderungen zu speichern, die Sie während dieser Übung vornehmen:

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

4. Melden Sie sich mit dem konfigurierten Benutzer, Passwort und der Master-API-URL bei OpenShift an.

```
[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. Erstellen Sie ein neues Projekt mit dem Namen `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. Erstellen Sie im OpenShift-Projekt `youruser-nodejs-app` eine neue Anwendung anhand des Quellcodes aus dem Verzeichnis `nodejs-app` im Git-Repository unter `https://github.com/yourgituser/D0180-apps`. Umgebungsvariable `npm_config_registry` verfügbar unter `http://${RHT_OCP4_NEXUS_SERVER}/repository/npm-proxy`. Benennen Sie die Anwendung `nodejs-dev`.

Der Build-Prozess für die Anwendung sollte fehlschlagen. Überwachen Sie den Build-Prozess, und identifizieren Sie den Build-Fehler.

- 6.1. Führen Sie den Befehl `oc new-app` zum Erstellen der Node.js-Anwendung aus.

```
[student@workstation ~]$ oc new-app --name nodejs-dev \
> https://github.com/${RHT_OCP4_GITHUB_USER}/D0180-apps#troubleshoot-review \
> -i nodejs:12 --context-dir=nodejs-app --build-env \
> npm_config_registry=http://${RHT_OCP4_NEXUS_SERVER}/repository/npm-proxy
--> Found image a2b5ec2 ...output omitted...

Node.js 12
...output omitted...
--> Creating resources ...
  imagestream.image.openshift.io "nodejs-dev" created
  buildconfig.build.openshift.io "nodejs-dev" created
  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. Überwachen Sie den Build-Fortschritt mit dem Befehl `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
```

Der Build-Prozess schlägt fehl, aus diesem Grund wird keine Anwendung ausgeführt. Im Build-Protokoll wird gezeigt, dass keine Version des express-Pakets vorhanden ist, das einer Versionsspezifikation von 4.20.x entspricht.

- 6.3. Führen Sie den Befehl `oc get pods` aus, um zu bestätigen, dass die Anwendung nicht bereitgestellt ist:

```
[student@workstation ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
nodejs-dev-1-build   0/1     Error      0          2m
```

7. Aktualisieren Sie die Version der Abhängigkeit `express` in der Datei `package.json` mit dem Wert `4.x`. Bestätigen und übertragen Sie die Änderungen an das Git-Repository.
- 7.1. Bearbeiten Sie die Datei `package.json` im Unterverzeichnis `nodejs-app`, und ändern Sie die Version der `express`-Abhängigkeit in `4.x`. Speichern Sie die Datei.

```
[student@workstation DO180-apps]$ cd nodejs-app
```

```
[student@workstation nodejs-app]$ sed -i s/4.20/4.x/ package.json
```

Die Datei enthält den folgenden Inhalt:

```
[student@workstation nodejs-app]$ cat package.json
{
  "name": "nodejs-app",
  "version": "1.0.0",
  "description": "Hello World App",
  "main": "server.js",
  "author": "Red{nbsp}Hat Training",
  "license": "ASL",
  "dependencies": {
    "express": "4.x",
    "html-errors": "latest"
  }
}
```

- 7.2. Übernehmen Sie die am Projekt vorgenommenen Änderungen und übertragen Sie sie.

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

Führen Sie im Terminalfenster den folgenden Befehl aus, um die Änderungen zu übernehmen und zu übertragen:

```
[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. Erstellen Sie die Anwendung neu. Verifizieren Sie, dass die Anwendung fehlerfrei erstellt wird.

- 8.1. Führen Sie den Befehl `oc start-build` aus, um die Anwendung neu zu erstellen.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-2 started
```

- 8.2. Führen Sie den Befehl `oc logs` aus, um die Build-Prozessprotokolle zu überwachen:

```
[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
```

Die Build-Erstellung ist erfolgreich, wenn ein Image an die interne OpenShift-Registry übertragen wird.

9. Verifizieren Sie, dass die Anwendung aufgrund eines Laufzeitfehlers nicht ausgeführt wird. Überprüfen Sie die Protokolle, und identifizieren Sie das Problem.

- 9.1. Führen Sie den Befehl `oc get pods` aus, um den Bereitstellungsstatus des Anwendungs-Pods zu überprüfen. Letztendlich sehen Sie, dass die erste Anwendungsbereitstellung den Status `CrashLoopBackoff` aufweist.

```
[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. Führen Sie den Befehl `oc logs -f deployment/nodejs-dev` aus, um die Protokolle für die Anwendungsbereitstellung nachzuvollziehen:

```
[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...
```

Das Protokoll zeigt an, dass die Datei `server.js` versucht, ein Modul mit dem Namen `http-error` zu laden. Die Variable `dependencies` in der Datei `packages` gibt an, dass der Modulname `html-errors` ist und nicht `http-error`.

10. Korrigieren Sie die Schreibweise der Abhängigkeit in der ersten Zeile der Datei `server.js`. Bestätigen und übertragen Sie die Änderungen an der Anwendung an das Git-Repository. Erstellen Sie die Anwendung neu. Verifizieren Sie nach dem Erstellen der Anwendung, dass die Anwendung ausgeführt wird.
 - 10.1. Korrigieren Sie die Schreibweise des Moduls in der ersten Zeile im `server.js` von `http-error` zu `html-errors`. Speichern Sie die Datei.

```
[student@workstation nodejs-app]$ sed -i s/http-error/html-errors/ server.js
```

Die Datei enthält den folgenden Inhalt:

```
[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. Übernehmen Sie die am Projekt vorgenommenen Änderungen und übertragen Sie sie.

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

```
[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. Führen Sie den Befehl `oc start-build` aus, um die Anwendung neu zu erstellen.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev
build.build.openshift.io/nodejs-dev-3 started
```

10.4. Führen Sie den Befehl `oc logs` aus, um die Build-Prozessprotokolle zu überwachen:

```
[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. Führen Sie den Befehl `oc get pods -w` aus, um die Bereitstellung der Pods für die Anwendung `nodejs-dev` zu überwachen:

```
[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
```

Nach dem dritten Build weist die zweite Bereitstellung den Status **Running** auf.

11. Erstellen Sie eine Route für die Anwendung, und testen Sie den Zugriff auf die Anwendung. Es sollte eine Fehlermeldung angezeigt werden. Überprüfen Sie die Protokolle, um den Fehler zu ermitteln.

11.1. Führen Sie den Befehl `oc expose` aus, um eine Route für die `nodejs-dev`-Anwendung zu erstellen:

```
[student@workstation nodejs-app]$ oc expose svc nodejs-dev
route.route.openshift.io/nodejs-dev exposed
```

11.2. Führen Sie den Befehl `oc get route` aus, um die URL der `nodejs-dev`-Route abzurufen:

```
[student@workstation nodejs-app]$ oc get route
NAME          HOST/PORT
nodejs-dev    nodejs-dev-your_user-nodejs-app.wildcard_domain ...
```

- 11.3. Verwenden Sie curl, um auf die Route zuzugreifen. Es sollte eine Fehlermeldung angezeigt werden.

```
[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. Überprüfen Sie die Protokolle für die nodejs-dev-Bereitstellung:

```
[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...
```

Der entsprechende Abschnitt der Datei server.js lautet:

```
app.get('/', function (req, res) {
  res.send('Hello World from pod: ' + process.env.HOSTNAME + '\n')
});
```

Ein process-Objekt in Node.js enthält einen Verweis auf ein env-Objekt und nicht auf ein environment-Objekt.

12. Ersetzen Sie zum Korrigieren des Fehlers process.environment durch process.env in der Datei server.js. Bestätigen und übertragen Sie die an der Anwendung vorgenommenen Änderungen an das Git-Repository. Erstellen Sie die Anwendung neu. Verifizieren Sie bei der Bereitstellung der neuen Anwendung, dass die Anwendung beim Zugriff auf die Anwendungs-URL keine Fehler generiert.

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

- 12.1. Ersetzen Sie zum Korrigieren des Fehlers `process.environment` durch `process.env` in der Datei `server.js`.

```
[student@workstation nodejs-app]$ sed -i \  
> s/process.environment/process.env/ server.js
```

Die Datei enthält den folgenden Inhalt:

```
[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. Übernehmen Sie die am Projekt vorgenommenen Änderungen und übertragen Sie sie.

```
[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. Führen Sie den Befehl `oc start-build` aus, um die Anwendung neu zu erstellen.

```
[student@workstation nodejs-app]$ oc start-build bc/nodejs-dev  
build.build.openshift.io/nodejs-dev-4 started
```

- 12.4. Führen Sie den Befehl `oc logs` aus, um die Build-Prozessprotokolle zu überwachen:

```
[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. Führen Sie den Befehl `oc get pods` aus, um die Bereitstellung der Pods für die Anwendung `nodejs-dev` zu überwachen:

```
[student@workstation nodejs-app]$ oc get pods  
NAME          READY   STATUS    RESTARTS   AGE  
nodejs-dev-1-build   0/1     Error      0          21m
```

Kapitel 8 | Fehlerbehebung bei containerisierten Anwendungen

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

Nach dem vierten Build weist die dritte Bereitstellung den Status **Running** auf.

- 12.6. Führen Sie den Befehl `curl` aus, um die Anwendung zu testen. Die Anwendung zeigt eine `Hello World`-Meldung an, die den Hostnamen des Anwendungs-Pods enthält:

```
[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
```

Bewertung

Bewerten Sie Ihre Arbeit, indem Sie auf Ihrem `workstation`-Rechner den Befehl `lab troubleshoot-review grade` ausführen. Beheben Sie sämtliche gemeldeten Fehler, und führen Sie das Skript so lange erneut aus, bis die Durchführung erfolgreich ist.

```
[student@workstation nodejs-app]$ lab troubleshoot-review grade
```

Beenden

Führen Sie auf `workstation` den Befehl `lab troubleshoot-review finish` aus, um diese praktische Übung abzuschließen.

```
[student@workstation nodejs-app]$ lab troubleshoot-review finish
```

Hiermit wird die praktische Übung abgeschlossen.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Anwendungen protokollieren normalerweise Aktivitäten wie Ereignisse, Warnungen und Fehler, um die Analyse des Anwendungsverhaltens zu erleichtern.
- Container-Anwendungen sollten Protokolldaten in eine Standardausgabe anstatt in einer Datei ausgeben, um den einfachen Zugriff auf Protokolle zu ermöglichen.
- Führen Sie den Befehl `podman logs` aus, um die Protokolle eines lokal mit Podman bereitgestellten Containers zu überprüfen.
- Führen Sie den Befehl `oc logs` aus, um auf die Protokolle für die Objekte `BuildConfig` und `Deployment` sowie auf einzelne Pods innerhalb eines OpenShift-Projekts zuzugreifen.
- Mit der Option `-f` können Sie die Protokollausgabe für die Befehle `podman logs` und `oc logs` nahezu in Echtzeit überwachen.
- Führen Sie den Befehl `oc port-forward` aus, um in einem Anwendungs-Pod eine direkte Verbindung zu einem Port herzustellen. Sie sollten diese Technik nur auf nicht in der Produktion befindliche Pods anwenden, da Interaktionen das Pod-Verhalten ändern können.

Kapitel 9

Ausführliche Wiederholung

Ziel

Wiederholen von Aufgaben aus *Red hat OpenShift I: Container & Kubernetes*

Ziele

- Wiederholen von Aufgaben aus *Red hat OpenShift I: Container & Kubernetes*

Abschnitte

- Ausführliche Wiederholung

Praktische Übung

- Ausführliche Überprüfung der Einführung in Container, Kubernetes und Red Hat OpenShift

Ausführliche Wiederholung

Ziele

In diesem Abschnitt werden das Wissen und die in *Red hat OpenShift I: Container & Kubernetes* vermittelten Kenntnisse überprüft und aufgefrischt.

Wiederholung von Red hat OpenShift I: Container & Kubernetes

Bevor Sie mit der ausführlichen Wiederholung für diesen Kurs beginnen, sollten Sie mit den in den jeweiligen Kapiteln behandelten Themen vertraut sein.

Für zusätzliche Übungen stehen Ihnen auch die vorherigen Kapitel dieses Lehrbuchs zur Verfügung.

Kapitel 1, Introducing Container Technology

Beschreiben, wie Anwendungen in von Red Hat OpenShift Container Platform orchestrierten Containern ausgeführt werden können.

- Beschreiben des Unterschieds zwischen Container-Anwendungen und herkömmlichen Bereitstellungen
- Beschreiben der Container-Architekturgrundlagen
- Beschreiben der Vorteile von Orchestrierungsanwendungen und von OpenShift Container Platform

Kapitel 2, Erstellen von containerisierten Services

Bereitstellen eines Services mit Container-Technologie

- Erstellen eines Datenbankservers anhand eines Container-Images

Kapitel 3, Verwalten von Containern

Ändern vordefinierter Container-Images zum Erstellen und Verwalten containerisierter Services

- Verwalten des Lebenszyklus eines Containers von der Erstellung bis zur Löschung
- Speichern der Container-Anwendungsdaten mit persistentem Storage
- Beschreiben, wie die Portweiterleitung für den Zugriff auf einen Container verwendet wird

Kapitel 4, Verwalten von Container-Images

Verwalten des Lebenszyklus eines Container-Images von der Erstellung bis zur Löschung

- Suchen und Abrufen von Images von Remote-Registries
- Exportieren, Importieren und Verwalten von Container-Images lokal und in einer Registry

Kapitel 5, Erstellen benutzerdefinierter Container-Images

Entwerfen und Codieren eines Containerfiles zum Erstellen eines benutzerdefinierten Container-Images

- Beschreiben der Ansätze zum Erstellen benutzerdefinierter Container-Images
- Erstellen eines Container-Images mit allgemeinen Containerfile-Befehlen

Kapitel 6, Bereitstellen containerisierter Anwendungen in OpenShift

Bereitstellen von Anwendungen in einem einzelnen Container in OpenShift Container Platform

- Beschreiben der Architektur von Kubernetes und von Red Hat OpenShift Container Platform.
- Erstellen von Kubernetes-Standardressourcen.
- Erstellen einer Route zu einem Service
- Erstellen einer Anwendung mit der Source-to-Image-Funktion in OpenShift Container Platform
- Erstellen einer Anwendung mit der OpenShift-Webkonsole

Kapitel 7, Bereitstellen von Anwendungen mit mehreren Containern

Bereitstellen von in Containern aufgeteilten Anwendungen mit mehreren Container-Images

- Beschreibung der Überlegungen zur Containerisierung von Anwendungen mit mehreren Container-Images
- Bereitstellen einer Anwendung mit mehreren Containern auf OpenShift Platform
- Bereitstellen einer Anwendung in OpenShift mithilfe einer Vorlage

Kapitel 8, Fehlerbehebung bei containerisierten Anwendungen

Fehlerbehebung bei containerisierten Anwendungen, die auf OpenShift bereitgestellt werden

- Fehlerbehebung bei Anwendungsbauten und Bereitstellungen in OpenShift
- Implementierung der Methoden zur Fehlerbehebung und für das Debugging bei containerisierten Anwendungen

► Praktische Übung

Containerisieren und Bereitstellen einer Softwareanwendung

In dieser Wiederholung containerisieren Sie einen Nexus-Server, erstellen und testen ihn mithilfe von Podman und stellen ihn in einem OpenShift-Cluster bereit.

Ergebnisse

Es werden folgende Fähigkeiten vermittelt:

- Schreiben eines Containerfiles zur erfolgreichen Containerisierung eines Nexus-Servers
- Erstellen eines Nexus-Server-Container-Images und Bereitstellung dieses mittels Podman
- Bereitstellung des Nexus-Server-Container-Images in einem OpenShift-Cluster

Bevor Sie Beginnen

Führen Sie das Setup-Skript für diese umfassende Wiederholung aus.

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

Die Lab-Dateien befinden sich im Verzeichnis `/home/student/D0180/labs/comprehensive-review`. Die Lösungsdateien befinden sich im Verzeichnis `/home/student/D0180/solutions/comprehensive-review`.

Anweisungen

Führen Sie die folgenden Schritte aus, um einen containerisierten Nexus-Server sowohl lokal als auch in OpenShift zu erstellen und zu testen:

1. Erstellen Sie ein Container-Image, das eine Instanz des Nexus-Servers startet:
 - Das Verzeichnis `/home/student/D0180/labs/comprehensive-review/image` enthält Dateien zum Erstellen des Containerimages. Führen Sie das Skript `get-nexus-bundle.sh` aus, um die Nexus-Serverdateien abzurufen.
 - Schreiben Sie ein Containerfile zur Containerisierung des Nexus-Servers. Das Containerfile muss sich im Verzeichnis `/home/student/D0180/labs/comprehensive-review/image` befinden. Das Containerfile muss zudem:
 - Das Basis-Image `ubi8/ubi:8.3` verwenden und einen willkürlichen Verwalter festlegen.
 - Die Umgebungsvariable `NEXUS_VERSION` auf `2.14.3-02` und `NEXUS_HOME` auf `/opt/nexus` festlegen.
 - Das Paket `java-1.8.0-openjdk-devel` installieren.
 - Einen Befehl ausführen, um einen `nexus`-Benutzer und eine -Gruppe zu erstellen. Beide verfügen über die UID und GID `1001`.

- Die Datei `nexus-2.14.3-02-bundle.tar.gz` im Verzeichnis `/${NEXUS_HOME}/` entpacken. `nexus-start.sh` zu demselben Verzeichnis hinzufügen.

Den Befehl `ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2` ausführen, um einen Symlink im Container zu erstellen. Einen Befehl ausführen, um die Inhaberschaft des Nexus-Benutzerverzeichnisses rekursiv in `nexus:nexus` zu ändern.

- Den Container als der Benutzer `nexus` ausführen und das Arbeitsverzeichnis auf `/opt/nexus` festlegen.
- Einen Volume-Mount-Punkt für das Container-Verzeichnis `/opt/nexus/sonatype-work` definieren. Der Nexus-Server speichert Daten in diesem Verzeichnis.
- Den standardmäßigen Container-Befehl auf `nexus-start.sh` festlegen.

Im Verzeichnis `/home/student/D0180/labs/comprehensive-review/image` befinden sich zwei `*.snippet`-Dateien, die die zur Erstellung des Nexus-Kontos und zur Installation von Java erforderlichen Befehle bereitstellen. Verwenden Sie die Dateien, um Sie beim Schreiben des Containerfiles zu unterstützen.

- Erstellen Sie das Container-Image mit dem Namen `nexus`.
- 2.** Erstellen und testen Sie das Container-Image mit Podman mithilfe einer Volume-Bereitstellung:
- Verwenden Sie das Skript `/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh`, um einen neuen Container mit einem Volume-Mount zu starten.
 - Überprüfen Sie die Container-Protokolle, um zu verifizieren, dass der Server gestartet ist und ausgeführt wird.
 - Testen Sie mithilfe der URL den Zugriff auf den Container-Service:
`http://127.0.0.1:18081/nexus`.
 - Entfernen Sie den Test-Container.
- 3.** Stellen Sie das Nexus-Server-Container-Image im OpenShift-Cluster bereit. Erforderliche Aktionen:
- Kennzeichnen Sie das Nexus-Server-Container-Image als `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest`, und übertragen Sie es per Push-Vorgang an die private Registry.
 - Erstellen Sie ein OpenShift-Projekt mit dem Namen `/${RHT_OCP4_DEV_USER}-review`.
 - Verarbeiten Sie die Vorlage `deploy/openshift/resources/nexus-template.json`, und erstellen Sie die Kubernetes-Ressourcen.
 - Erstellen Sie eine Route für den Nexus-Service. Verifizieren Sie, dass Sie über `workstation` auf `http://nexus-`${RHT_OCP4_DEV_USER}-review`${RHT_OCP4_WILDCARD_DOMAIN}/nexus/` zugreifen können.

Bewertung

Nachdem Sie das Nexus-Server-Container-Image im OpenShift-Cluster bereitgestellt haben, verifizieren Sie Ihre Arbeit, indem Sie das Lab-Auswertungsskript ausführen:

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

Beenden

Führen Sie auf workstation den Befehl `lab comprehensive-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

► Lösung

Containerisieren und Bereitstellen einer Softwareanwendung

In dieser Wiederholung containerisieren Sie einen Nexus-Server, erstellen und testen ihn mithilfe von Podman und stellen ihn in einem OpenShift-Cluster bereit.

Ergebnisse

Es werden folgende Fähigkeiten vermittelt:

- Schreiben eines Containerfiles zur erfolgreichen Containerisierung eines Nexus-Servers
- Erstellen eines Nexus-Server-Container-Images und Bereitstellung dieses mittels Podman
- Bereitstellung des Nexus-Server-Container-Images in einem OpenShift-Cluster

Bevor Sie Beginnen

Führen Sie das Setup-Skript für diese umfassende Wiederholung aus.

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

Die Lab-Dateien befinden sich im Verzeichnis /home/student/D0180/labs/comprehensive-review. Die Lösungsdateien befinden sich im Verzeichnis /home/student/D0180/solutions/comprehensive-review.

Anweisungen

Führen Sie die folgenden Schritte aus, um einen containerisierten Nexus-Server sowohl lokal als auch in OpenShift zu erstellen und zu testen:

1. Erstellen Sie ein Container-Image, das eine Instanz des Nexus-Servers startet:
 - Das Verzeichnis /home/student/D0180/labs/comprehensive-review/image enthält Dateien zum Erstellen des Containerimages. Führen Sie das Skript `get-nexus-bundle.sh` aus, um die Nexus-Serverdateien abzurufen.
 - Schreiben Sie ein Containerfile zur Containerisierung des Nexus-Servers. Das Containerfile muss sich im Verzeichnis /home/student/D0180/labs/comprehensive-review/image befinden. Das Containerfile muss zudem:
 - Das Basis-Image `ubi8/ubi:8.3` verwenden und einen willkürlichen Verwalter festlegen.
 - Die Umgebungsvariable `NEXUS_VERSION` auf `2.14.3-02` und `NEXUS_HOME` auf `/opt/nexus` festlegen.
 - Das Paket `java-1.8.0-openjdk-devel` installieren.
 - Einen Befehl ausführen, um einen `nexus`-Benutzer und eine -Gruppe zu erstellen. Beide verfügen über die UID und GID `1001`.

Kapitel 9 | Ausführliche Wiederholung

- Die Datei `nexus-2.14.3-02-bundle.tar.gz` im Verzeichnis `${NEXUS_HOME} /` entpacken. `nexus-start.sh` zu demselben Verzeichnis hinzufügen.

Den Befehl `ln -s ${NEXUS_HOME}/nexus-$NEXUS_VERSION ${NEXUS_HOME}/nexus2` ausführen, um einen Symlink im Container zu erstellen. Einen Befehl ausführen, um die Inhaberschaft des Nexus-Benutzerverzeichnisses rekursiv in `nexus:nexus` zu ändern.

- Den Container als der Benutzer `nexus` ausführen und das Arbeitsverzeichnis auf `/opt/nexus` festlegen.
- Einen Volume-Mount-Punkt für das Container-Verzeichnis `/opt/nexus/sonatype-work` definieren. Der Nexus-Server speichert Daten in diesem Verzeichnis.
- Den standardmäßigen Container-Befehl auf `nexus-start.sh` festlegen.

Im Verzeichnis `/home/student/D0180/labs/comprehensive-review/image` befinden sich zwei `*.snippet`-Dateien, die die zur Erstellung des Nexus-Kontos und zur Installation von Java erforderlichen Befehle bereitstellen. Verwenden Sie die Dateien, um Sie beim Schreiben des Containerfiles zu unterstützen.

- Erstellen Sie das Container-Image mit dem Namen `nexus`.

- 1.1. Führen Sie das Skript `get-nexus-bundle.sh` aus, um die Nexus-Serverdateien abzurufen.

```
[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. Schreiben Sie ein Containerfile zur Containerisierung des Nexus-Servers. Rufen Sie das Verzeichnis `/home/student/D0180/labs/comprehensive-review/image` auf, und erstellen Sie das Containerfile.

- Legen Sie das zu verwendende Basis-Image fest:

```
FROM ubi8/ubi:8.3
```

- Geben Sie einen beliebigen Namen und eine E-Mail-Adresse als Verwalter ein:

```
FROM ubi8/ubi:8.3  
MAINTAINER username <username@example.com >
```

- Legen Sie ein Build-Argument für `NEXUS_VERSION` und eine Umgebungsvariable für `NEXUS_HOME` fest:

```
FROM ubi8/ubi:8.3  
MAINTAINER username <username@example.com>  
  
ARG NEXUS_VERSION=2.14.3-02  
ENV NEXUS_HOME=/opt/nexus
```

Kapitel 9 | Ausführliche Wiederholung

- Installieren Sie das Paket `java-1.8.0-openjdk-devel` mit dem Befehlyum.

```
FROM ubi8/ubi:8.3
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
```

- Erstellen Sie das Serverstartverzeichnis und das Servicekonto bzw. die -gruppe. Erstellen Sie das Benutzerverzeichnis so, dass es dem Servicekonto angehört.

```
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}
```

- Führen Sie den Container als Benutzer `nexus` aus.

```
USER nexus
```

- Installieren Sie die Nexus-Server Software unter `NEXUS_HOME`, und fügen Sie das Startskript hinzu. Beachten Sie, dass die ADD-Richtlinie die Nexus-Dateien extrahieren wird.

```
ADD nexus-${NEXUS_VERSION}-bundle.tar.gz ${NEXUS_HOME}
ADD nexus-start.sh ${NEXUS_HOME}/
```

- Erstellen Sie den symbolischen Link `nexus2`, der auf das Verzeichnis des Nexus-Servers verweist. Ändern Sie die Inhaberschaft des Verzeichnisses `${NEXUS_HOME}` rekursiv in `nexus:nexus`.

```
RUN ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} \
${NEXUS_HOME}/nexus2
```

- Legen Sie `/opt/nexus` als aktuelles Arbeitsverzeichnis fest:

```
WORKDIR ${NEXUS_HOME}
```

- Definieren Sie einen Mount-Punkt für das Volume, um die persistenten Daten des Nexus-Servers zu speichern:

```
VOLUME ["/opt/nexus/sonatype-work"]
```

- Legen Sie die Anweisung CMD auf die Skriptdatei `nexus-start.sh` fest.

CMD ["sh", "nexus-start.sh"]

Das abgeschlossene Containerfile sollte wie folgt lauten:

```
FROM ubi8/ubi:8.3

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. Erstellen Sie das Container-Image mit dem Namen nexus.

```
[student@workstation image]$ podman build --layers=false -t nexus .
STEP 1: FROM ubi8/ubi:8.3
Getting image source signatures
...output omitted...
STEP 14: COMMIT ...output omitted...localhost/nexus:latest
...output omitted...
```

2. Erstellen und testen Sie das Container-Image mit Podman mithilfe einer Volume-Bereitstellung:

- Verwenden Sie das Skript `/home/student/D0180/labs/comprehensive-review/deploy/local/run-persistent.sh`, um einen neuen Container mit einem Volume-Mount zu starten.
- Überprüfen Sie die Container-Protokolle, um zu verifizieren, dass der Server gestartet ist und ausgeführt wird.

Kapitel 9 | Ausführliche Wiederholung

- Testen Sie mithilfe der URL den Zugriff auf den Container-Service:
`http://127.0.0.1:18081/nexus`.
 - Entfernen Sie den Test-Container.
- 2.1. Führen Sie das Skript `run-persistent.sh` aus. Ersetzen Sie den Namen des Containers entsprechend der Ausgabe des Befehls `podman ps`.
- ```
[student@workstation images]$ cd /home/student/D0180/labs/comprehensive-review
[student@workstation comprehensive-review]$ cd deploy/local
[student@workstation local]$./run-persistent.sh
80970007036bbb313d8eeb7621fada0ed3f0b4115529dc50da4dccef0da34533
```
- 2.2. Überprüfen Sie die Container-Protokolle, um zu verifizieren, dass der Server gestartet ist und ausgeführt wird.
- ```
[student@workstation local]$ 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. Führen Sie den Befehl `curl` aus, um den Container mithilfe der URL zu testen:
`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. Entfernen Sie den Test-Container.
- ```
[student@workstation local]$ podman rm -f inspiring_poincare  
81f480f21d475af683b4b003ca6e002d37e6aaa581393d3f2f95a1a7b7eb768b
```
3. Stellen Sie das Nexus-Server-Container-Image im OpenShift-Cluster bereit. Erforderliche Aktionen:
- Kennzeichnen Sie das Nexus-Server-Container-Image als `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest`, und übertragen Sie es per Push-Vorgang an die private Registry.
 - Erstellen Sie ein OpenShift-Projekt mit dem Namen `${RHT_OCP4_DEV_USER} - review`.
 - Verarbeiten Sie die Vorlage `deploy/openshift/resources/nexus-template.json`, und erstellen Sie die Kubernetes-Ressourcen.
 - Erstellen Sie eine Route für den Nexus-Service. Verifizieren Sie, dass Sie über `workstation` auf `http://nexus-${RHT_OCP4_DEV_USER} - review ${RHT_OCP4_WILDCARD_DOMAIN}/nexus/` zugreifen können.

Kapitel 9 | Ausführliche Wiederholung

- 3.1. Melden Sie sich bei Ihrem Quay.io-Benutzerkonto an.

```
[student@workstation local]$ podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password: your_quay_password
Login Succeeded!
```

- 3.2. Veröffentlichen Sie das Nexus-Server-Container-Image in Ihrer quay.io-Registry.

```
[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. Repositorys, die durch das Übertragen von Images per Push-Vorgang an quay.io erstellt wurden, sind standardmäßig privat. Details zum Ändern der Repository-Sichtbarkeit finden Sie im Abschnitt **Sichtbarkeit von Repositorys** von Anhang C.

- 3.4. Erstellen Sie das OpenShift-Projekt:

```
[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. Ersetzen Sie RHT_OCP4_QUAY_USER in der Ressourcendatei durch Ihren Quay-Benutzernamen, und erstellen Sie die Kubernetes-Ressourcen:

```
[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. Stellen Sie den Service bereit, indem Sie eine Route erstellen:

```
[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. Stellen Sie über einen Browser unter `http://nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/` eine Verbindung zur Webanwendung des Nexus-Servers her.

Bewertung

Nachdem Sie das Nexus-Server-Container-Image im OpenShift-Cluster bereitgestellt haben, verifizieren Sie Ihre Arbeit, indem Sie das Lab-Auswertungsskript ausführen:

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

Beenden

Führen Sie auf `workstation` den Befehl `lab comprehensive-review finish` aus, um diese praktische Übung abzuschließen.

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

Hiermit wird die praktische Übung abgeschlossen.

Anhang A

Implementieren der Microservices-Architektur

Ziel

Refactoring einer Anwendung in Microservices.

Ziele

- Aufteilen einer Anwendung auf mehrere Container zum Trennen bestimmter Ebenen und Services.

Abschnitte

- Implementieren der Microservices-Architektur (und angeleitete Übung)

Implementieren der Microservices-Architektur

Ziele

In diesem Abschnitt werden die folgenden Themen behandelt:

- Aufteilen einer Anwendung auf mehrere Container zum Trennen bestimmter Ebenen und Services.
- Beschreiben typischer Ansätze zum Zerlegen einer monolithischen Anwendung in mehrere bereitstellbare Einheiten.
- Beschreiben, wie die Anwendung To Do List in drei Container aufgeteilt werden kann, die mit ihren logischen Stufen übereinstimmen.

Vorteile der Aufteilung einer monolithischen Anwendung in Container

Bei der herkömmlichen Anwendungsentwicklung werden für gewöhnlich viele bestimmte Funktionen als einzelne Bereitstellungseinheit oder als monolithische Anwendung gebündelt. Im Rahmen der klassischen Bereitstellung werden zudem möglicherweise unterstützende Services, beispielsweise Datenbanken und andere Middleware-Service, auf demselben Server wie die Anwendung bereitgestellt. Obwohl monolithische Anwendungen weiterhin in einem Container bereitgestellt werden können, sind viele Vorteile einer Container-Architektur, wie Skalierbarkeit und Flexibilität, nicht so weit verbreitet. Das Auflösen von Monolithen erfordert sorgfältige Überlegungen. Es wird empfohlen, dass in Microservices-Anwendungen jeder Microservice die Mindestfunktionalität ausführt, die für jeden Container isoliert ausgeführt werden kann.

Kleinere Container und das Zerlegen einer Anwendung und deren unterstützende Services in mehrere Container bieten viele Vorteile, beispielsweise:

- Höhere Hardwareauslastung, da sich die kleineren Container leichter in die verfügbare Hostkapazität einpassen lassen.
- Einfachere Skalierung, da Teile der Anwendung skaliert werden können, um einen erhöhten Workload zu unterstützen, ohne dass andere Teile der Anwendung skaliert werden.
- Einfachere Upgrades, da Entwickler Teile der Anwendung aktualisieren können, ohne dass davon andere Teile derselben Anwendung betroffen sind.

Zwei verbreitete Möglichkeiten zum Zerlegen einer Anwendung:

- Stufen: basiert auf Architekturebenen.
- Services: basiert auf Anwendungsfunktionen.

Aufteilen auf Basis von Ebenen (Stufen)

Anwendungen werden durch Entwickler oftmals in Stufen eingeteilt. Diese Einteilung basiert darauf, wie nah die Funktionen an den Endbenutzern und wie weit weg sie von Datenspeichern sind. Ein gutes Beispiel der klassischen 3-Stufen-Architektur sind die Darstellung, Geschäftslogik und Persistenz.

Diese logische Architektur entspricht in der Regel einer physischen Bereitstellungsarchitektur, in der eine Darstellungsebene auf einem Webserver, die Geschäftsebene auf einem Anwendungsserver und die Persistenzebene auf einem Datenbankserver bereitgestellt wird.

Durch das Zerlegen einer Anwendung in Stufen können sich Entwickler anhand der Stufen der Anwendung auf bestimmte Technologien spezialisieren. Einige Entwickler konzentrieren sich zum Beispiel auf Webanwendungen, während andere die Datenbankentwicklung bevorzugen. Ein weiterer Vorteil besteht in der Fähigkeit, alternative Stufenimplementierungen auf Grundlage unterschiedlicher Technologien bereitzustellen. Dazu zählt beispielsweise das Erstellen einer Mobilanwendung als ein anderes Front-End für eine vorhandene Anwendung. Die Mobilanwendung würde als alternative Darstellungsstufe fungieren und die Geschäfts- und Persistenzstufen der ursprünglichen Webanwendung wiederverwenden.

In kleineren Anwendungen werden die Darstellungs- und Unternehmensstufen für gewöhnlich als eine einzelne Einheit bereitgestellt. Das kann beispielsweise auf demselben Webserver sein. Mit zunehmender Auslastung wird die Darstellungsebene jedoch in ihre eigene Bereitstellung verschoben, um die Auslastung zu verteilen. In kleineren Anwendungen wird möglicherweise sogar die Datenbank eingebettet. Anspruchsvollere Anwendungen werden oftmals auf diese monolithische Art von Entwicklern erstellt und bereitgestellt.

Wenn Entwickler eine monolithische Anwendung in Stufen aufteilen, nehmen sie normalerweise mehrere Änderungen vor:

- Verbindungsparameter zu Datenbanken und anderen Middleware-Services, beispielsweise Messaging, wurden in festen IP-Adressen oder Hostnamen hartcodiert, für gewöhnlich `localhost`. Sie müssen in Parameter aufgeteilt werden, um auf externe Server zu verweisen, die sich möglicherweise von der Entwicklung zur Produktion voneinander unterscheiden.
- Im Falle von Webanwendungen ist es nicht möglich, Ajax-Aufrufe mit relativen URLs vorzunehmen. Sie müssen eine absolute URL verwenden, die auf einen festen öffentlichen DNS-Hostnamen verweist.
- Moderne Webbrowser lehnen als Sicherheitsmaßnahme Ajax-Aufrufe zu Servern ab, die sich von dem im Skript enthaltenen unterscheiden, das den Aufruf vornimmt. Die Anwendung benötigt die Berechtigungen für die Ressourcenfreigabe zwischen verschiedenen Ursprüngen (Cross-Origin Resource Sharing, CORS).

Nachdem die Anwendungsebenen aufgeteilt wurden, sodass sie auf unterschiedlichen Servern ausgeführt werden können, sollte es kein Problem sein, sie in unterschiedlichen Containern auszuführen.

Aufteilen auf Basis von diskreten Services

Die meisten komplexen Anwendungen bestehen aus vielen teilautonomen Services. Zum Beispiel weist ein Onlinestore einen Produktkatalog, Warenkorb, Zahlungs- und Versandoptionen usw. auf.

Wenn sich die Leistung eines bestimmten Services in einer monolithischen Anwendung verschlechtert, impliziert die Skalierung des Services zum Verbessern der Leistung, dass die Leistung von allen anderen konstituierenden Anwendungsservices verbessert wird. Wenn der verschlechterte Service jedoch Teil einer Microservices-Architektur ist, wird der betroffene Service unabhängig von den anderen Anwendungsservices skaliert. Die folgende Abbildung zeigt die Serviceskalierung sowohl für eine monolithisch- als auch für einen Microservices-basierte Architektur:

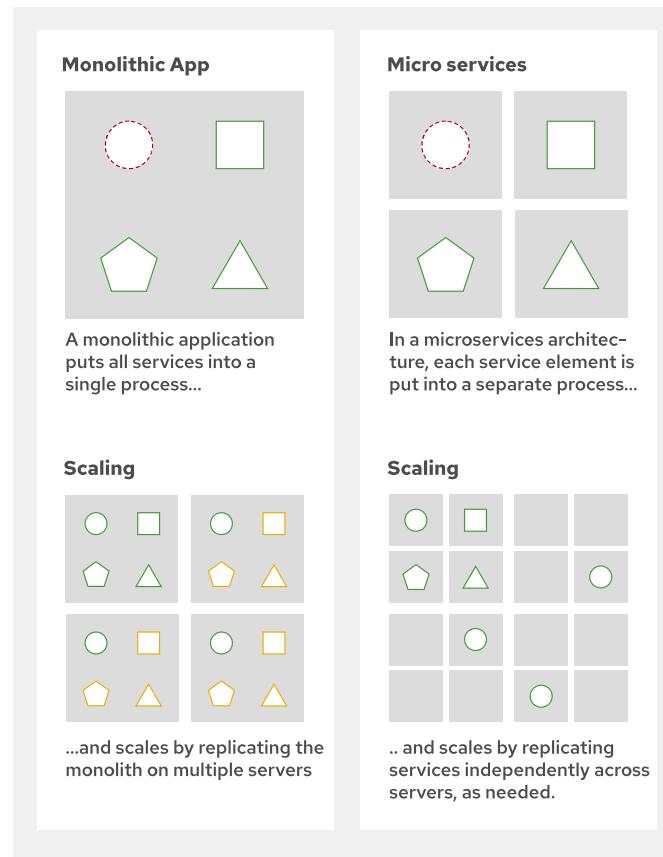


Abbildung A.1: Vergleich der Anwendungsskalierung in einer monolithischen Architektur mit einer Microservices-Architektur

Sowohl traditionelle serviceorientierte Architekturen (SOA) als auch neuere Microservices-Architekturen bündeln diese Funktionsgruppen zu eigene Einheiten und stellen sie als solche bereit. Dadurch kann jeder Funktionssatz durch ein eigenes Team entwickelt, aktualisiert und skaliert werden, ohne andere Funktionssätze (oder Services) zu beeinträchtigen. Funktionsübergreifende Anteile wie die Authentifizierung können zudem als Services, die von anderen Serviceimplementierungen verwendet werden, gebündelt und bereitgestellt werden.

Das Aufteilen jedes Anteils in einen separaten Server hat möglicherweise viele Anwendungen zur Folge. Sie werden logisch aufgebaut, gebündelt und als eine kleine Anzahl an Einheiten bereitgestellt. Manchmal verwendet sogar eine einzelne monolithische Einheit einen Serviceansatz.

Container ermöglichen die Materialisierung von Architekturen auf der Basis von Services während der Bereitstellung. Aus diesem Grund treten Microservices und Container normalerweise zusammen auf. Container allein sind jedoch nicht genug. Sie müssen von Orchestrierungstools ergänzt werden, um die Abhängigkeiten unter den Services zu verwalten.

Die Microservices-Architektur verwendet servicebasierte Architekturen bis zum Äußersten. Ein Service ist so klein wie möglich (ohne Aufteilung eines Funktionssatzes). Er wird als eine unabhängige Einheit bereitgestellt und verwaltet, anstatt Teil einer größeren Anwendung zu sein. Dadurch können vorhandene Microservices wiederverwendet werden, um neue Anwendungen zu erstellen.

Zum Aufteilen einer Anwendung in Services ist dieselbe Art von Änderung wie beim Aufteilen in Stufen erforderlich. Parametrisieren Sie beispielsweise Verbindungsparameter in Datenbanken und andere Middleware-Services, und kümmern Sie sich um Webbrowser-Sicherheitsfunktionen.

Refactoring der Anwendung „To Do List“

To Do List ist eine einfache Anwendung mit einem einzelnen Funktionssatz, deren Zerlegung in Services nicht wirklich aussagekräftig ist. Beim Refactoring in Darstellungs- und Unternehmensstufen, d. h. in ein Front-End und ein Back-End, um sie in bestimmten Containern bereitzustellen, wird jedoch derselbe Typ an Änderungen veranschaulicht, die für eine typische Anwendung erforderlich wären, um in Services aufgeteilt zu werden.

In der folgenden Abbildung wird gezeigt, wie die Anwendung To Do List in drei Containern bereitgestellt werden würde, wobei einer für jede Ebene steht:

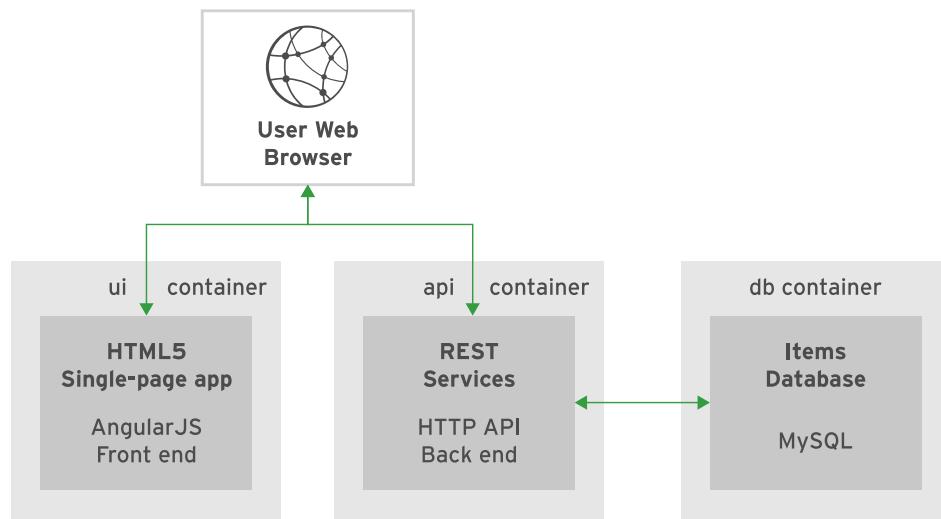


Abbildung A.2: In Ebenen aufgeteilte Anwendung „To Do List“, wobei jede Ebene als Container bereitgestellt ist

Der Vergleich zwischen dem Quellcode der ursprünglichen monolithischen Anwendung und der refaktorierten bietet einen Überblick über die Änderungen:

- Das Front-End-JavaScript-Element in `script/items.js` verwendet „`workstation.lab.example.com`“ als Hostnamen für den Zugriff auf das Back-End.
- Das Back-End verwendet Umgebungsvariablen zum Abrufen der Datenbankverbindungsparameter.
- Das Back-End muss auf Anforderungen antworten und verwendet dabei das Verb HTTP `OPTIONS` mit Headern, die den Webbrowser anweisen, Anforderungen zu akzeptieren, die von unterschiedlichen DNS-Domains mit CORS stammen.

Andere Versionen des Back-End-Services weisen möglicherweise ähnliche Änderungen auf. Jede Programmiersprache und jedes REST-Framework besitzt ihre bzw. seine eigene(n) Syntax und Features.



Anmerkung

Seite zu monolithischen Anwendungen in Wikipedia [https://en.wikipedia.org/wiki/Monolithic_application]

CORS-Seite in Wikipedia [https://en.wikipedia.org/wiki/Cross-origin_resource_sharing]

► Angeleitete Übung

Refactoring der Anwendung „To Do List“

In diesem Lab strukturieren Sie die Anwendung To Do List so um, dass sie in mehrere Container aufgeteilt wird, die miteinander verbunden sind. Dadurch können die HTML 5-Front-End-Anwendung, die Node.js-REST-API und der MySQL-Server in ihren eigenen Containern ausgeführt werden.

Ergebnisse

Sie sollten in der Lage sein, eine monolithische Anwendung in ihre Ebenen zu refaktorieren und jede Stufe als einen Microservice bereitzustellen.

Bevor Sie Beginnen

Führen Sie den folgenden Befehl aus, um die Arbeitsverzeichnisse für das Lab mit den Dateien für die Anwendung To Do List einzurichten:

```
[student@workstation ~]$ lab appendix-microservices start
```

Anweisungen

► 1. Verschieben der HTML-Dateien

Der erste Schritt beim Refactoring der Anwendung To Do List besteht darin, den Front-End-Code aus der Anwendung in seinen eigenen aktiven Container zu verschieben. In diesem Schritt verschieben Sie die HTML-Anwendung und die abhängigen Dateien in ihr eigenes Verzeichnis für die Bereitstellung auf einem Apache-Server, der in einem Container ausgeführt wird.

- 1.1. Verschieben Sie die HTML- und statischen Dateien aus der monolithischen Node.js-Anwendung To Do List in das Verzeichnis `src/`:

```
[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. Die aktuelle Front-End-Anwendung interagiert mit einer relativen URL mit dem API-Service. Da die API und der Front-End-Code nun in separaten Containern ausgeführt werden, muss das Front-End so angepasst werden, dass es auf die absolute URL der API für die Anwendung To Do List verweist.

Öffnen Sie die Datei `/home/student/D0180/labs/appendix-microservices/apps/html5/src/script/item.js`. Suchen Sie am Ende der Datei nach der folgenden Methode:

```
app.factory('itemService', function ($resource) {
  return $resource('api/items/:id');
});
```

Ersetzen Sie den Code durch folgenden Inhalt:

```
app.factory('itemService', function ($resource) {
    return $resource("http://workstation.lab.example.com:30080/todo/api/items/:id");
});
```

Stellen Sie sicher, dass die neue URL keine Zeilenumbrüche enthält. Speichern Sie die Datei, und beenden Sie den Editor.

► 2. Erstellen des HTML/Images

- 2.1. Melden Sie sich mit Ihrem Red Hat-Benutzerkonto beim Red Hat Container Catalog an.

```
[student@workstation html5]$ podman login registry.redhat.io
Username: your_username
Password: your_password
Login Succeeded!
```

- 2.2. Erstellen Sie das untergeordnete Apache-Image:

```
[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. Verifizieren Sie, ob das Image richtig erstellt wurde:

```
[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. Ändern der REST-API zum Verbinden mit externen Containern

- 3.1. Die REST-API verwendet derzeit hartcodierte Werte zum Herstellen einer Verbindung zur MySQL-Datenbank. Bearbeiten Sie die Datei `/home/student/D0180/labs/appendix-microservices/apps/nodejs/models/db.js`, in der die Datenbankkonfiguration enthalten ist. Aktualisieren Sie die `Datenbankname` , `Nutzername` , und `Passwort` Werte, um stattdessen Umgebungsvariablen zu verwenden. Aktualisieren Sie zudem `params.host` so, dass auf den Hostnamen des Hosts verwiesen wird, auf dem der MySQL-Container ausgeführt wird. Aktualisieren

Sie `params.port` so, dass der umgeleitete Port zum Container berücksichtigt wird. Beide Werte sind als die Umgebungsvariablen `MYSQL_SERVICE_HOST` bzw. `MYSQL_SERVICE_PORT` verfügbar. Ersetzte Inhalte sollten folgendermaßen aussehen:

```
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'
  }
};
```



Anmerkung

Diese Datei kann über `/home/student/D0180/solutions/appendix-microservices/apps/nodejs/models/db.js` kopiert und eingefügt werden.

- 3.2. Konfigurieren Sie das Back-End, um die Ressourcenfreigabe zwischen verschiedenen Ursprüngen (Cross-Origin Resource Sharing, CORS) zu verarbeiten. Dazu kommt es, wenn eine Ressourcenanforderung von einer anderen Domain erfolgt als der, in der die Anforderung gestellt wurde. Da die API Anforderungen von einer anderen DNS-Domain (die Front-End-Anwendung) verarbeiten muss, müssen Sicherheitsausnahmen erstellt werden, damit diese Anforderungen erfolgreich sind. Nehmen Sie die folgenden Änderungen an der Anwendung in der von Ihnen gewünschten Sprache vor, um mit CORS zu arbeiten.

Fügen Sie "restify-cors-middleware": "1.1.1" als neue Abhängigkeit zur Datei `package.json` unter `/home/student/D0180/labs/appendix-microservices/apps/nodejs/package.json` hinzu. Setzen Sie ein Komma an das Ende der vorherigen Abhängigkeit. Stellen Sie sicher, dass das Ende der Datei wie folgt aussieht:

```
"sequelize": "5.21.1",
"mysql2": "2.0.0",
"restify-cors-middleware": "1.1.1"
}
```

Aktualisieren Sie die Datei `app.js` unter `/home/student/D0180/labs/appendix-microservices/apps/nodejs/app.js`, um die CORS-Verwendung zu konfigurieren. Fordern Sie das Modul `restify-cors-middleware` in der zweiten Zeile an, und aktualisieren Sie dann den Inhalt der Datei so, dass sie dem Folgenden entspricht:

```
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);
```

„origins“ mit dem Wert ["*"] weisen den Server an, alle Domains zuzulassen. In einem Produktionsserver wäre dieser Wert in der Regel ein Array von Domains, die bekanntermaßen Zugriff auf die API benötigen.

► 4. Erstellen des REST-API-Images

- 4.1. Erstellen Sie das untergeordnete REST-API-Image mit dem folgenden Befehl. Von diesem Image wird das Node.js-Image verwendet.

```
[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. Führen Sie den Befehl `podman images` aus, um zu verifizieren, dass alle erforderlichen Images erfolgreich erstellt wurden:

```
[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. Ausführen der Container

- 5.1. Verwenden Sie das Skript `run.sh`, um die Container auszuführen:

```
[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. Führen den Befehl `podman ps` aus, um zu bestätigen, dass alle drei Container ausgeführt werden:

```
[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. Testen der Anwendung

- 6.1. Verwenden Sie den Befehl `curl`, um zu verifizieren, dass die REST-API für die Anwendung To Do List richtig funktioniert:

```
[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. Öffnen Sie Firefox auf dem Rechner `workstation`, und navigieren Sie zu 127.0.0.1:30000. Dort sollte die Anwendung To Do List angezeigt werden.
- 6.3. Wechseln Sie zurück in den Benutzerordner des Benutzers.

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

Beenden

Führen Sie auf `workstation` das Skript `lab appendix-microservices finish` aus, um diese praktische Übung zu beenden.

```
[student@workstation ~]$ lab appendix-microservices finish
```

Hiermit ist die angeleitete Übung beendet.

Zusammenfassung

In diesem Kapitel wurden die folgenden Themen behandelt:

- Das Aufteilen einer monolithischen Anwendung in mehrere Container ermöglicht eine bessere Anwendungsskalierbarkeit, erleichtert Upgrades und ermöglicht eine höhere Hardwareauslastung.
- Die drei allgemeinen Ebenen für die logische Trennung einer Anwendung sind die Darstellungs-, die Geschäfts- und die Persistenzebene.
- Die Ressourcenfreigabe zwischen verschiedenen Ursprüngen (Cross-Origin Resource Sharing, CORS) kann Ajax-Aufrufe an Server verhindern, die sich von denen unterscheiden, von denen die Seiten heruntergeladen wurden. Stellen Sie sicher, dass Vorkehrungen getroffen werden, um CORS aus anderen Containern in der Anwendung zuzulassen.
- Container-Images sind so konzipiert, dass sie unveränderbar sind. Konfigurationen können jedoch zum Zeitpunkt der Image-Erstellung oder durch das Erstellen von persistentem Storage für Konfigurationen weitergegeben werden.

Anhang B

Erstellen eines GitHub-Benutzerkontos

Ziel

Beschreiben, wie ein GitHub-Benutzerkonto für praktische Übungen im Kurs erstellt wird

Erstellen eines GitHub-Benutzerkontos

Ziele

Nach Abschluss dieses Abschnitts sollten Sie in der Lage sein, ein GitHub-Benutzerkonto sowie öffentliche Git-Repositorys für die Labs im Kurs zu erstellen.

Erstellen eines GitHub-Benutzerkontos

Sie benötigen ein GitHub-Benutzerkonto, um ein oder mehrere *öffentliche* Git-Repositorys für die Labs in diesem Kurs zu erstellen. Falls Sie bereits ein GitHub-Benutzerkonto besitzen, können Sie die in diesem Anhang aufgelisteten Schritte überspringen.



Wichtig

Falls Sie bereits ein GitHub-Benutzerkonto besitzen, sollten Sie sicherstellen, dass Sie für die Labs in diesem Kurs nur *öffentliche* Git-Repositorys erstellen. Die Übungsskripts und -anweisungen zur Bewertung der Labs erfordern authentifizierte Zugriff, um das Repository zu klonen. Es muss möglich sein, ohne Passwörter, SSH-Schlüssel oder GPG-Schlüssel auf die Repositorys zuzugreifen.

Führen Sie die folgenden Schritte aus, um ein neues GitHub-Benutzerkonto zu erstellen:

1. Navigieren Sie mit einem Webbrowser zu <https://github.com>.
2. Geben Sie die erforderlichen Details ein, und klicken Sie dann auf **Sign up for GitHub**.

+

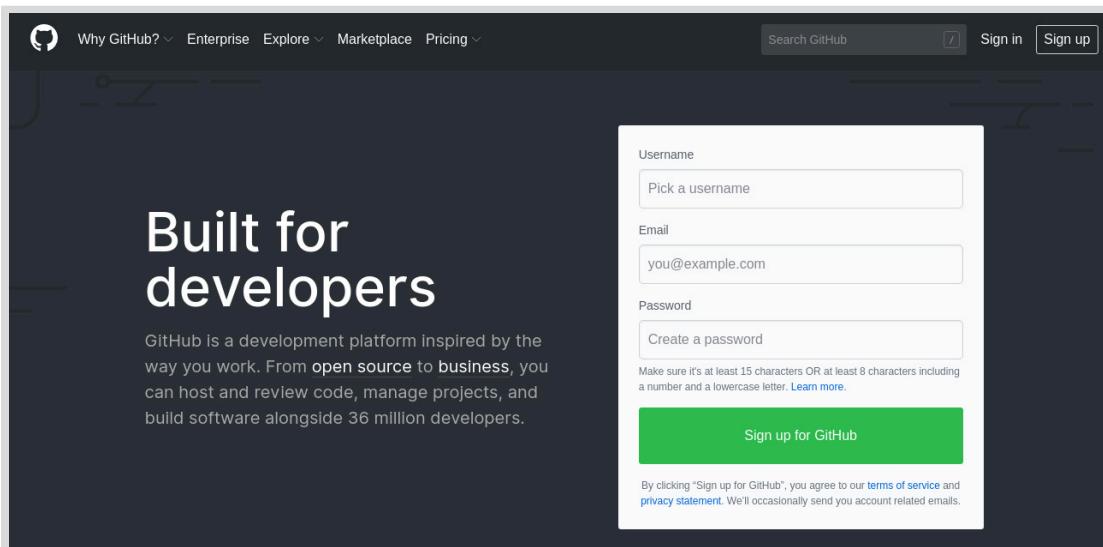


Abbildung B.1: Erstellen eines GitHub-Benutzerkontos

1. Sie erhalten eine E-Mail mit Anweisungen zur Aktivierung Ihres GitHub-Benutzerkontos. Verifizieren Sie Ihre E-Mail-Adresse, und melden Sie sich dann während der

Benutzerkontoerstellung auf der GitHub-Website mit dem Benutzernamen und dem Passwort an.

2. Nachdem Sie sich bei GitHub angemeldet haben, können Sie neue Git-Repositorys erstellen. Klicken Sie dazu im Bereich **Repositories** auf der linken Seite der GitHub-Startseite auf **New**.

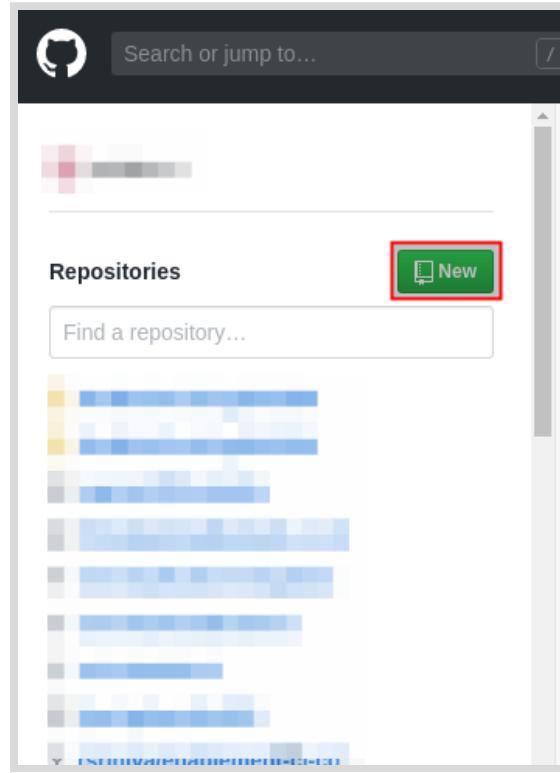


Abbildung B.2: Erstellen eines neuen Git-Repositorys

Alternativ können Sie auf das Pluszeichen (+) in der rechten oberen Ecke (rechts neben dem Glockensymbol) und dann auf [New repository] klicken.

+

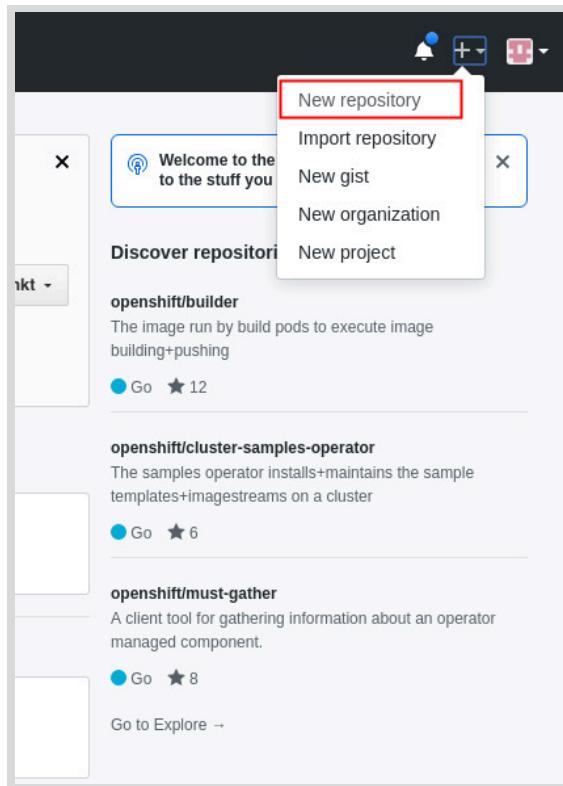


Abbildung B.3: Erstellen eines neuen Git-Repositorys



Literaturhinweise

Registrieren eines neuen GitHub-Benutzerkontos

<https://help.github.com/en/articles/signing-up-for-a-new-github-account>

Anhang C

Erstellen eines Quay-Benutzerkontos

Ziel

Beschreiben, wie ein Quay-Benutzerkonto für praktische Übungen im Kurs erstellt wird

Erstellen eines Quay-Benutzerkontos

Ziele

Nach Abschluss dieses Abschnitts sollten Sie in der Lage sein, ein Quay-Benutzerkonto sowie öffentliche Container-Image-Repositorys für die praktischen Übungen im Kurs zu erstellen.

Erstellen eines Quay-Benutzerkontos

Sie benötigen ein Quay-Benutzerkonto, um ein oder mehrere *öffentliche* Container-Image-Repositorys für die praktischen Übungen in diesem Kurs zu erstellen. Wenn Sie bereits ein Quay-Benutzerkonto besitzen, können Sie die Schritte zum Erstellen eines neuen Benutzerkontos überspringen, die in diesem Anhang aufgeführt sind.



Wichtig

Wenn Sie bereits über ein Quay-Benutzerkonto verfügen, achten Sie darauf, dass Sie nur *öffentliche* Container-Image-Repositorys für die praktischen Übungen in diesem Kurs erstellen. Die Übungsskripts und -anweisungen zur Bewertung der praktischen Übungen erfordern authentifizierte Zugriff, um Container-Images aus dem Repository abzurufen.

Führen Sie die folgenden Schritte aus, um ein neues Quay-Benutzerkonto zu erstellen:

1. Navigieren Sie mit einem Webbrowser zu <https://quay.io>.
2. Klicken Sie in der rechten oberen Ecke (neben der Suchleiste) auf **Sign in**.
3. Auf der Seite **Sign in** können Sie sich mit Ihren Red Hat- (erstellt in Anhang D), Google- oder GitHub-Anmeldedaten (erstellt in Anhang A) anmelden.

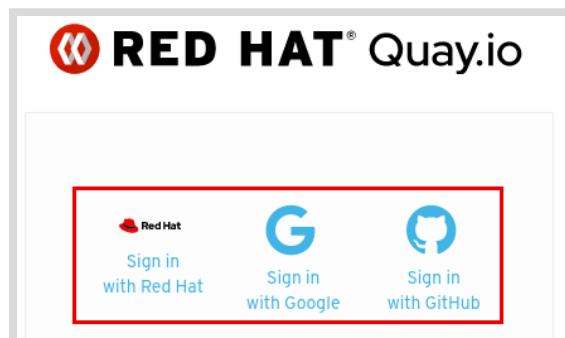


Abbildung C.1: Melden Sie sich mit Google- oder GitHub-Anmeldedaten an.

Alternativ können Sie auf **Create Account** klicken, um ein neues Benutzerkonto zu erstellen.

Anhang C | Erstellen eines Quay-Benutzerkontos

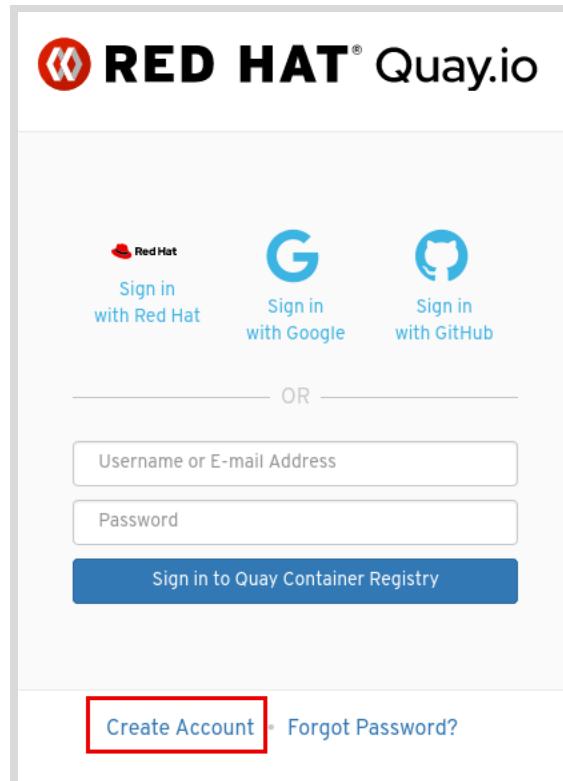


Abbildung C.2: Erstellen eines neuen Kontos

1. Wenn Sie die Anmeldemethode über Red Hat, Google oder GitHub nicht nutzen und sich stattdessen für die Erstellung eines neuen Benutzerkontos entschieden haben, erhalten Sie eine E-Mail mit Anweisungen zur Aktivierung Ihres Quay-Benutzerkontos. Überprüfen Sie Ihre E-Mail-Adresse und melden Sie sich dann während der Kontoerstellung auf der Quay-Website mit dem Benutzernamen und dem Passwort an.
2. Nachdem Sie sich bei Quay angemeldet haben, können Sie neue Image-Repositorys erstellen, indem Sie auf der Seite **Repositories** auf **Create New Repository** klicken.

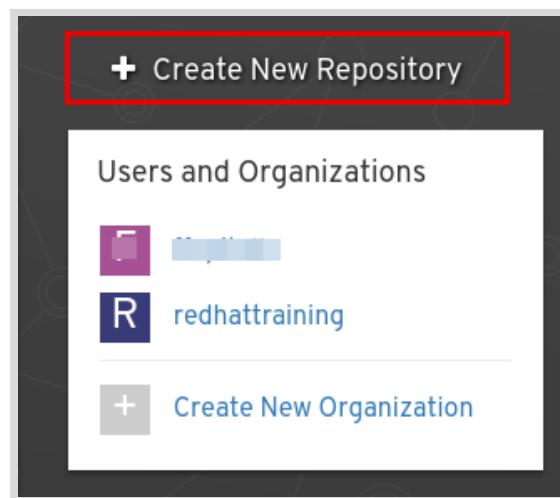


Abbildung C.3: Erstellen eines neuen Image-Repositorys

Alternativ können Sie auf das Pluszeichen (+) in der rechten oberen Ecke (links neben dem Glockensymbol) und dann auf **New Repository** klicken.

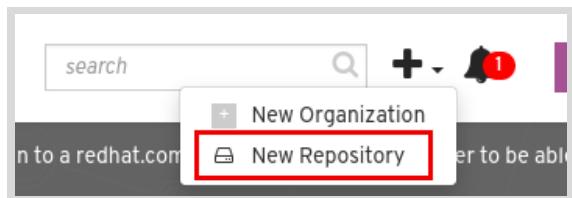


Abbildung C.4: Erstellen eines neuen Image-Repositorys

Arbeiten mit CLI-Tools

Wenn Sie Ihr Benutzerkonto mit Red Hat, Google oder GitHub erstellt haben, müssen Sie ein Benutzerkontopasswort festlegen, um CLI-Tools wie Podman oder Docker zu verwenden.

1. Klicken Sie oben rechts auf <YOUR_USERNAME>.
2. Klicken Sie auf **Account Settings**.
3. Klicken Sie auf den Link *Change password*.



Literaturhinweise

Erste Schritte mit Quay.io

<https://docs.quay.io/solution/getting-started.html>

Sichtbarkeit von Repositorys

Ziele

Nach Abschluss dieses Abschnitts sollten Sie in der Lage sein, die Repository-Sichtbarkeit auf Quay.io zu steuern.

Sichtbarkeit von Quay.io-Repositorys

Mit Quay.io können öffentliche und private Repositorys erstellt werden. Öffentliche Repositorys können von jedem ohne Einschränkungen gelesen werden, obwohl Schreibberechtigungen explizit erteilt werden müssen. Bei privaten Repositorys sind die Lese- und Schreibberechtigungen eingeschränkt. Die Anzahl der privaten Repositorys in quay.io ist jedoch in Abhängigkeit des Namespace-Plans begrenzt.

Standardsichtbarkeit von Repositorys

Repositorys, die durch das Übertragen von Images per Push-Vorgang an quay.io erstellt wurden, sind standardmäßig privat. Damit OpenShift (oder ein anderes Tool) diese Images abrufen kann, können Sie in OpenShift und Quay einen Private Key konfigurieren oder das Repository auf öffentlich festlegen, damit keine Authentifizierung erforderlich ist. Die Einrichtung von Private Keys wird in diesem Dokument nicht behandelt.

The screenshot shows the Quay.io interface with the URL https://quay.io/repository/. At the top, there is a navigation bar with back, forward, and refresh buttons, followed by a lock icon and the URL. Below the navigation bar, the Red Hat logo is displayed next to the text "RED HAT® Quay.io". To the right of the logo are links for "EXPLORE", "APPLICATIONS", "REPOSITORIES" (which is highlighted in red), and "TUTORIAL". The main content area has a dark background with a network-like graphic. The title "Repositories" is centered at the top of the content area. Below it, a section titled "Starred" is shown, indicating that no repositories have been starred yet. A message says "You haven't starred any repositories yet." and "Stars allow you to easily access your favorite repositories." There are four repository cards listed under the "RHT_OCP4_QUAY_USER" heading:

- do180-mysql-57-rhel7
- do180-todonodejs
- do180-quote-php
- nexus

Each card includes a small icon, the repository name, and a star icon for favoriting.

Aktualisieren der Repository-Sichtbarkeit

Um die Repository-Sichtbarkeit auf öffentlich festzulegen, wählen Sie das entsprechende Repository in <https://quay.io/repository/> (melden Sie sich bei Bedarf bei Ihrem Benutzerkonto an) aus, und öffnen Sie die Seite **Settings**. Klicken Sie dazu unten links auf das Zahnrad. Scrollen Sie nach unten zum Abschnitt **Repository Visibility**, und klicken Sie auf die Schaltfläche „Make Public“.

The screenshot shows the Quay.io repository settings interface. At the top, there is a message "Trust Disabled" with a gear icon and a button to "Enable Trust". Below this, under "Events and Notifications", it says "No notifications have been setup for this repository." with a "Create Notification" button. Under "Repository Visibility", it indicates the repository is "private" and has a lock icon. A button to "Make Public" is shown. At the bottom, there is a "Delete Repository" section with a warning message: "Deleting a Repository cannot be undone. Here be dragons!" and a red "Delete Repository" button.

Kehren Sie zur Liste der Repositorys zurück. Das Sperrsymbol neben dem Repository-Namen ist verschwunden. Das Repository ist nun also öffentlich.

Anhang D

Erstellen eines Red Hat-Benutzerkontos

Ziel

Beschreiben, wie ein Red Hat-Benutzerkonto für praktische Übungen im Kurs erstellt wird

Erstellen eines Red Hat-Benutzerkontos

Ziele

Nach Abschluss dieses Abschnitts sollten Sie in der Lage sein, ein Red Hat-Benutzerkonto für den Zugriff auf die Red Hat Container Catalog-Images für die praktischen Übungen im Kurs zu erstellen.

Erstellen eines Red Hat-Benutzerkontos

Sie benötigen ein Red Hat-Benutzerkonto, um auf die Red Hat Container Catalog-Images zuzugreifen. Falls Sie bereits ein Red Hat-Benutzerkonto besitzen, können Sie diese Schritte überspringen.

Führen Sie die folgenden Schritte aus, um ein neues Red Hat-Benutzerkonto zu erstellen:

1. Navigieren Sie mit einem Webbrowser zu <https://redhat.com>.
2. Klicken Sie oben rechts auf **Log in**.
3. Klicken Sie im rechten Fensterbereich auf **Register now**.

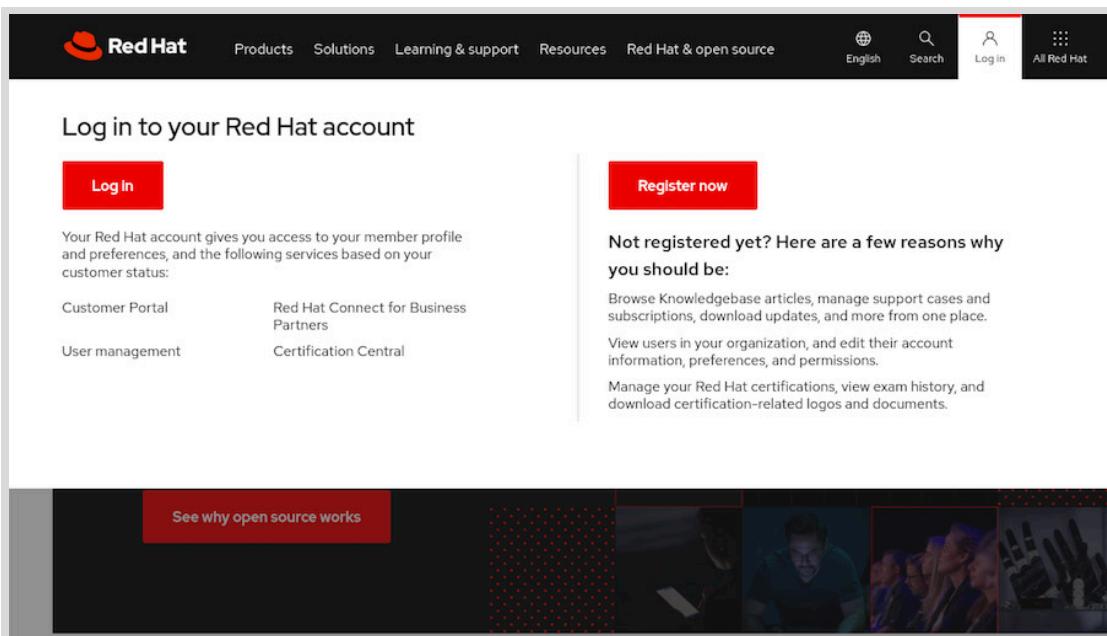


Abbildung D.1: Registrieren Sie ein neues Benutzerkonto.

4. Legen Sie Account type auf Personal fest.

Anhang D | Erstellen eines Red Hat-Benutzerkontos

The screenshot shows the 'Create a Red Hat account' page. At the top is the Red Hat logo with a red hat icon. Below it is the heading 'Create a Red Hat account'. A sub-instruction says 'Sign up and use this Red Hat account to access all of Red Hat's applications, communities, support, and more.' A note below states: 'Red Hat will collect your contact and account information to create your Red Hat account. We use your personal data to identify you and to provide you with information, support, and customer service. For more information, please see Red Hat's [privacy statement](#)'. A section titled 'Join an existing account' explains how to access subscriptions through an existing account or contact customer service. A note at the bottom right indicates '* Required fields'. The 'Account type' section asks to choose between 'Corporate' and 'Personal'. The 'Personal' option is selected, with a description: 'A personal Red Hat account is for purchasing or administering your own personal systems.'

Abbildung D.2: Legen Sie den Kontotyp fest.

5. Füllen Sie den Rest des Formulars mit Ihren persönlichen Daten aus. In diesem Formular wählen Sie auch den Benutzernamen und das Passwort für dieses Benutzerkonto aus.
6. Klicken Sie auf **CREATE MY ACCOUNT**.

The screenshot shows the 'CREATE MY ACCOUNT' form. It has two input fields: 'City *' with a placeholder 'Enter city' and 'State/Province *' with a placeholder 'Enter state/province'. Below the fields is a large red button labeled 'CREATE MY ACCOUNT'. At the bottom is a link '« Back to login page'.

Abbildung D.3: Füllen Sie das Formular mit den persönlichen Daten aus.



Literaturhinweise

Red Hat Customer Portal

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

Anhang E

Nützliche Git-Befehle

Ziel

Beschreiben der nützlichen Git-Befehle, die für die praktischen Übungen in diesem Kurs verwendet werden

Git-Befehle

Ziele

Nach Abschluss dieses Abschnitts sollten Sie in der Lage sein, die Übungen in diesem Kurs neu zu starten und zu wiederholen. Sie sollten ferner in der Lage sein, von einer unvollständigen Übung zu einer anderen zu wechseln, und später die vorherige Übung da fortzusetzen, wo Sie aufgehört haben.

Arbeiten mit Git-Zweigen

In diesem Kurs wird ein auf GitHub gehostetes Git-Repository verwendet, um den Anwendungsquellcode des Kurses zu speichern. Zu Beginn des Kurses erstellen Sie einen eigenen Fork dieses Repository, das auch auf GitHub gehostet wird.

Sie arbeiten in diesem Kurs mit einer lokalen Kopie Ihres Fork, den Sie auf die `workstation`-VM klonen. Der Begriff `origin` bezieht sich auf das Remote-Repository, von dem ein lokales Repository geklont wird.

Für die Übungen im Kurs verwenden Sie jeweils einen separaten Git-Zweig. Alle Änderungen, die Sie am Quellcode vornehmen, erfolgen in einem neuen Branch, den Sie nur für diese Übung erstellen. Nehmen Sie niemals Änderungen am `master`-Branch vor.

Nachfolgend finden Sie eine Liste von Szenarien und die entsprechenden Git-Befehle, die Sie verwenden können, um mit Branches zu arbeiten und eine Wiederherstellung eines als fehlerfrei bekannten Zustands durchzuführen.

Wiederholen einer Übung von Anfang an

Führen Sie die folgenden Schritte aus, um eine bereits abgeschlossene Übung von Anfang an zu wiederholen:

1. Im Rahmen der Übung übergeben und übertragen Sie alle Änderungen in Ihren lokalen Branch. Sie haben die Übung abgeschlossen, indem Sie den Unterbefehl `finish` ausgeführt haben, um alle Ressourcen zu bereinigen:

```
[student@workstation ~]$ `lab _your-exercise_ finish`
```

2. Navigieren Sie zu Ihrem lokalen Klon des Repositorys `D0180-apps`, und wechseln Sie zum Branch `master`:

```
[student@workstation ~]$ `cd ~/D0180-apps`  
[student@workstation D0180-apps]$ `git checkout master`
```

3. Löschen Sie Ihren lokalen Branch:

```
[student@workstation D0180-apps]$ `git branch -d _your-branch_`
```

4. Löschen Sie den Remote-Branch in Ihrem persönlichen GitHub-Benutzerkonto:

Anhang E | Nützliche Git-Befehle

```
[student@workstation D0180-apps]$ `git push origin --delete _your-branch_`
```

5. Führen Sie den Unterbefehl `start` aus, um die Übung neu zu starten:

```
[student@workstation D0180-apps]$ `cd ~`  
[student@workstation ~]$ `lab _your-exercise_ start`
```

Abbrechen einer teilweise abgeschlossenen Übung und Neustarten von Anfang an

Es kann vorkommen, dass Sie einige Schritte in der Übung teilweise abgeschlossen haben und den aktuellen Versuch abbrechen und von Anfang an neu beginnen möchten. Führen Sie die folgenden Schritte aus:

1. Führen Sie den Unterbefehl `finish` der Übung aus, um alle Ressourcen zu bereinigen.

```
[student@workstation ~]$ `lab _your-exercise_ finish`
```

2. Wechseln Sie zu Ihrem lokalen Klon des Repositorys D0180-apps, und verwerfen Sie mit `git stash` alle ausstehenden Änderungen im aktuellen Branch:

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

3. Wechseln Sie zum `master`-Branch Ihres lokalen Repositorys:

```
[student@workstation D0180-apps]$ `git checkout master`
```

4. Löschen Sie Ihren lokalen Branch:

```
[student@workstation D0180-apps]$ `git branch -d _your-branch_`
```

5. Löschen Sie den Remote-Branch in Ihrem persönlichen GitHub-Benutzerkonto:

```
[student@workstation D0180-apps]$ `git push origin --delete _your-branch_`
```

6. Sie können die Übung nun neu starten, indem Sie den Unterbefehl `start` ausführen:

```
[student@workstation D0180-apps]$ `cd ~`  
[student@workstation ~]$ `lab _your-exercise_ start`
```

Wechseln zu einer anderen Übung aus einer unvollständigen Übung

Es kann vorkommen, dass Sie einige Schritte in einer Übung teilweise abgeschlossen haben, aber Sie möchten zu einer anderen Übung wechseln und die aktuelle Übung zu einem späteren Zeitpunkt erneut aufrufen.

Vermeiden Sie es, zu viele Übungen unabgeschlossen zu belassen, um sie später erneut zu bearbeiten. Diese Übungen binden Cloud-Ressourcen und Sie könnten Ihr zugewiesenes

Anhang E | Nützliche Git-Befehle

Kontingent auf dem Cloud-Anbieter und auf dem OpenShift-Cluster aufbrauchen, den Sie gemeinsam mit anderen Kursteilnehmern verwenden. Wenn Sie der Meinung sind, dass es möglicherweise eine Weile dauert, bis Sie zur aktuellen Übung zurückkehren können, sollten Sie diese Aufgabe abbrechen und später neu starten.

Wenn Sie die aktuelle Übung unterbrechen und an der nächsten Übung arbeiten möchten, führen Sie die folgenden Schritte aus:

- Übergeben Sie alle ausstehenden Änderungen in Ihrem lokalen Repository und übertragen Sie diese an Ihr persönliches GitHub-Benutzerkonto. Möglicherweise möchten Sie den Schritt aufzeichnen, in dem Sie die Übung angehalten haben:

```
[student@workstation ~]$ `cd ~/DO180-apps`  
[student@workstation DO180-apps]$ `git commit -a -m 'Paused at step X.Y'`  
[student@workstation DO180-apps]$ `git push`
```

- Führen Sie den Befehl `finish` der ursprünglichen Übung nicht aus. Dies ist wichtig, um Ihre bestehenden OpenShift-Projekte unverändert zu belassen, sodass Sie später fortfahren können.
- Starten Sie die nächste Übung, indem Sie den Unterbefehl `start` ausführen:

```
[student@workstation ~]$ `lab _your-exercise_ start`
```

- Die nächste Übung wechselt zum `master`-Branch und erstellt optional einen neuen Branch für Änderungen. Das bedeutet, dass die Änderungen, die an der ursprünglichen Übung im ursprünglichen Branch vorgenommen wurden, unverändert bleiben.
- Wenn Sie nach Abschluss der nächsten Übung zur ursprünglichen Übung zurückkehren möchten, wechseln Sie zu ihrem Branch zurück:

```
[student@workstation ~]$ `git checkout _original-branch_`
```

Anschließend können Sie mit der ursprünglichen Übung bei dem Schritt fortfahren, bei dem Sie aufgehört haben.



Anmerkung

Manpage Git branch [<https://git-scm.com/docs/git-branch>]

What is a Git branch? [<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>]