DevOps Culture and Practice

Facilitator Guide

DO500-OCP3.11-en-1-20190228

Table of Contents

OCP 3.11 DO500
DevOps Culture and Practice
Edition 1 20190228

Authors: Richard Allred, Jim Rigsbee, Ravishankar Srinivasan
Editor: Seth Kenlon

Derived from materials created by Red Hat Open Innovation Labs

Introduction

Red Hat OpenShift Development I: Containerizing Applications

Red Hat OpenShift Container Platform, based on container technology and Kubernetes, provides developers an enterprise-ready solution for developing and deploying containerized software applications.

Red Hat OpenShift Development I: Containerizing Applications (DO288), the second course in the OpenShift development track, teaches students how to design, build, and deploy containerized software applications on an OpenShift cluster. Whether writing native container applications or migrating existing applications, this course provides hands-on training to boost developer productivity powered by Red Hat OpenShift Container Platform.

Objectives

- Design, build, and deploy containerized applications on an OpenShift cluster.

Audience

- Software Developers
- Software Architects

Prerequisites

- Either have completed the *Introduction to Containers, Kubernetes, and Red Hat OpenShift* (DO180) course, or have equivalent knowledge.
- A minimum of an RHCSA certification or equivalent experience to navigate and use the command line and perform shell scripting

Chapter 1: Deploying and Managing Applications on an OpenShift Cluster

Overview

Goal	Deploy applications using various application packaging methods to an OpenShift cluster and manage their resources.
Objectives	• Deploy an application to the cluster from a Dockerfile with the CLI. • Deploy an application from a container image and manage its resources using the web console. • Deploy an application from source code and manage its resources using the command line interface.
Sections	• Deploying an Application to an OpenShift Cluster (and Guided Exercise) • Managing Applications and the Web Console (and Guided Exercise) • Managing Applications with the CLI (and Guided Exercise)
Lab	Deploying and Managing Applications on an OpenShift Cluster

Deploying an Application to an OpenShift Cluster

Objectives

After completing this section, students should be able to:

- Deploy an application to the cluster from a Dockerfile with the CLI.
- Describe the resources created in a project by the **oc new-app** command and the web console.

Development Paths

1. Deployment Scenarios

- Red Hat OpenShift Container Platform is designed for building and deploying containerized applications:
  - Complete application life cycle is managed using OpenShift tools
  - Existing containerized applications are built outside of OpenShift
- **oc new-app** command creates resources to build (optional) and deploy

2. Building and Deploying with CLI

- **oc new-app** command:
  - Pass a URL that points to a Git repository or a container image
- Command decides what type of application to build and/or deploy:
  - Source-to-Image
  - Dockerfile
  - Docker Image

3. Specifying the Builder Image

- Specifying a builder image stream on the Git URL:

```
oc new-app php~http://gitserver.example.com/mygitrepo
```

- Specifying a builder image stream with the image stream argument:

```
oc new-app -i php http://gitserver.example.com/mygitrepo
```

4. Deploying Existing Containerized Applications

- Specify the registry and image name:

```
oc new-app registry.example.com/mycontainerimage
```

5. Deploying Existing Dockerfiles

- Specify the Git repository that contains the Dockerfile

```
oc new-app http://gitserver.example.com/mydockerfilerepo
```

6. Disambiguation Options for **oc new-app**

Option	Description
--image-stream or **i**	Provides the image stream to be used as either the S2I builder image for an S2I build or to deploy a container image.
--strategy	Provides the build strategy, usually **docker** or **source**
--code	Provides the URL to a Git repository to be used as input to an S2I build.
--docker-image	Provides the URL to a container image to be deployed.

7. Generated Resources

- Resources created by **oc new-app** command:
  - Build Configuration
  - Image Stream
  - Deployment Configuration
  - Service for all ports exposed
- These resources create other Kubernetes resources

8. Image Streams

- An *image stream* resource refers to container images and caches their metadata
- Build and Deployment configurations use image stream events to trigger behavior:
  - Trigger a new S2I build when the builder image changes
  - Trigger a new deployment when the application container image changes
- To create a new image stream use the **oc import-image** command:

```
oc import-image myis --confirm \
   --from registry.acme.example.com:5000/acme/awesome --insecure
```

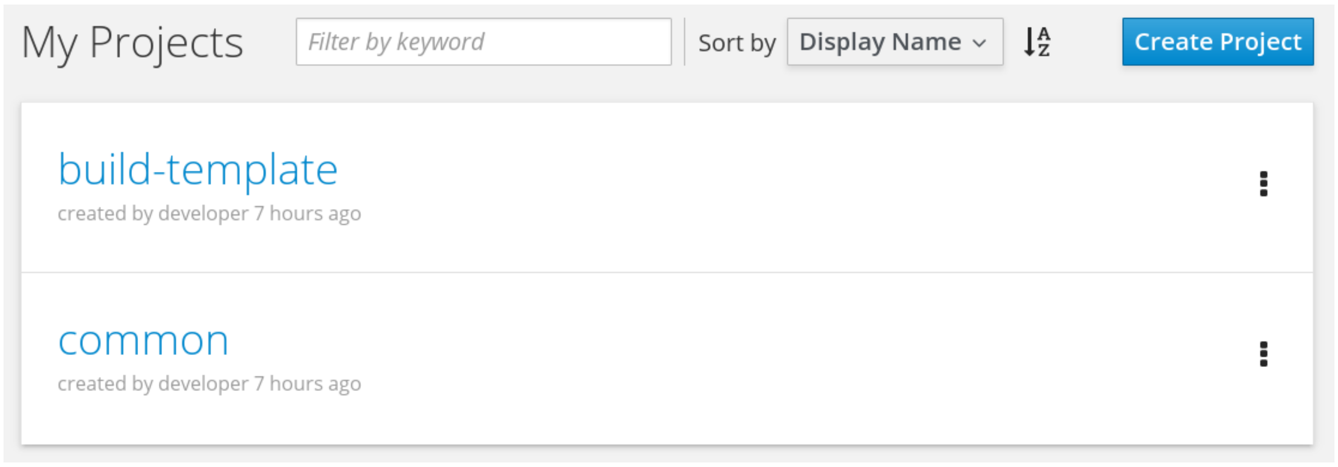

Figure 1. Web console projects listing

9. Image Stream Tags

- An *image stream tag* points to a particular container image in an image stream
  - **ruby:2.0** refers to **openshift3/ruby-20-rhel7**
  - **ruby:2.2** refers to **openshift3/ruby-22-rhel7**
- An image stream can have one or more tags
- Note web console projects listing in figure Web console projects listing

References

Further information is available in the Application Life Cycle Management chapter of the *Developer Guide* for Red Hat OpenShift Container Platform 3.6; at https://access.redhat.com/documentation/en-us/openshift_container_platform/3.6/html/developer_guide/

Guided Exercise: Deploying an Application to an OpenShift Cluster

In this exercise, you will use OpenShift to build and deploy an application from a Dockerfile.

NOTE Make sure you perform the clean up tasks at the end of each exercise.

Outcomes

You should be able to create an application using the **docker** build strategy, and delete all resources from the application without deleting the project.

Before you begin

To perform this exercise, you need to ensure you have access to:

- A running OpenShift cluster
- The parent image for the sample application (**rhel7:7.3**)
- The sample application Git repository (**rhel7-echo**)

Run the following command on the **workstation** VM to validate the prerequisites and to download solution files:

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

Steps

1. Inspect the application Dockerfile project.

   a. Clone the project from the classroom Git server to the **student** user's home folder:

   ```
   [student@workstation ~]$ git clone http://services.lab.example.com/rhel7-echo
   ```

   b. Review the Dockerfile for the application:

   ```
   [student@workstation ~]$ cat ~/rhel7-echo/Dockerfile
   FROM registry.lab.example.com/rhel7:7.3  ①
   CMD bash -c "while true; do echo test; sleep 5; done"  ②
   ```

   ① The parent image is the base OS image for Red Hat Enterprise Linux (RHEL) 7.3 from the Red Hat Container Catalog (RHCC).

   ② The application runs a loop that echoes "test" every five seconds.

2. Build the application container image using the OpenShift cluster.

a. Log in to OpenShift as the **developer** user:

```
[student@workstation ~]$ oc login -u developer -p redhat \
     https://master.lab.example.com
```

b. Create a new project for the application:

```
[student@workstation ~] oc new-project docker-build
```

c. Create a new application from the Dockerfile project:

```
[student@workstation ~] oc new-app --name echo --insecure-registry \
    http://services.lab.example.com/rhel7-echo
...
--> Creating resources ...
    imagestream "rhel7" created
    imagestream "echo" created
    buildconfig "echo" created
    deploymentconfig "echo" created
--> Success
...
```

Ignore the warnings about failed registry lookups and image stream matches. They are part of the algorithm to determine which resources to create.

The output shows that the **oc new-app** command correctly identified the URL as a Git repository and created a build configuration.

d. Follow the build logs:

```
[student@workstation ~]$ oc logs -f bc/echo
Cloning "http://services.lab.example.com/rhel7-echo" ...
...
Pulling image registry.lab.example.com:5000/rhel7:7.3 ...
...
Step 1 : FROM registry.lab.example.com:5000/rhel7:7.3 ①
...
Step 2 : CMD bash -c "while true; do echo test; sleep 15; done"
...
Step 3 : ENV "OPENSHIFT_BUILD_NAME" "echo-1" ... ②
...
Successfully built 434df2dba1bd
Pushing image docker-registry.default.svc:5000/docker-build/echo:latest ...
...
Push successful
```

① The **oc new-app** command correctly identified the Git repository as a Dockerfile project and the

OpenShift build performs a Dockerfile build.

② OpenShift adds metadata to the application container image using ENV and LABEL instructions.

3. Verify that the application works inside OpenShift.

remainder of lab has been removed for brevity of example

Summary

In this chapter, you learned that:

- OpenShift can run container images created externally. Container images created using OpenShift can also be used externally.
- OpenShift…

Chapter 2: Designing Containerized Applications for OpenShift

Overview

Goal	Select an application containerization method for an application and package it to run on an OpenShift cluster.
Objectives	• Select an appropriate application containerizations method. • Build a container image with advanced Dockerfile directives. • Select a method for injecting configuration data into an application and create the necessary resources to do so.
Sections	• Selecting a Containerization Approach (and Quiz) • Building Container Images with Advanced Dockerfile Directives (and Guided Exercise) • Injecting Configuration Data into an Application (and Guided Exercise)
Lab	Designing Containerized Applications for OpenShift

Selecting a Containerization Approach

Objective

After completing this section, students should be able to select an appropriate application containerization method.

Selecting a Build Method

1. Methods of Containerization

- There are several ways to create container images:
  - Container Images built outside of OpenShift
  - Dockerfiles
  - Source-to-Image builder images containing:
    - based operating system libraries
    - compilers and interpreters
    - run times
    - frameworks
    - Source-to-Image tooling

Remainder of section has been removed

References

Red Hat Container Catalog

https://access.redhat.com/documentation/en-us/openshift_container_platform/3.6/html/developer_guide/

Dockerfiles for images that are part of the Red Hat Software Collections library are available at

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

Further information about ...

Quiz: Selecting a Containerization Approach

Choose the correct answers to the following questions:

1. You have been asked to deploy a commercial, third-party, .NET-based application to an OpenShift cluster, which is packaged by the vendor as a container image. Which of the following options would you use to deploy the application?
   a. A Source-to-Image build.
   b. A custom Source-to-Image builder.
   c. Stage the container image in a private docker registry, and then deploy the container image to an OpenShift cluster using the OpenShift **oc** command-line tool.
   d. None of these. You cannot deploy .NET-based applications on an OpenShift cluster
2. You are tasked…

Quiz Solution

1. You have been asked to deploy a commercial, third-party, .NET-based application to an OpenShift cluster, which is packaged by the vendor as a container image. Which of the following options would you use to deploy the application?
   a. ~~A Source-to-Image build.~~
   b. ~~A custom Source-to-Image builder.~~
   c. Stage the container image in a private docker registry, and then deploy the container image to an OpenShift cluster using the OpenShift **oc** command-line tool.
   d. ~~None of these. You cannot deploy .NET-based applications on an OpenShift cluster.~~
2. You are tasked…